

REFERENCE ONLY



NAVAL UNDERWATER SYSTEMS CENTER  
NEW LONDON LABORATORY  
NEW LONDON, CONNECTICUT 06320

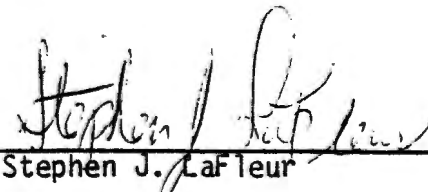
Technical Memorandum

SIMAS ADM XBT ALGORITHM

REFERENCE ONLY

Date: 5 DEC 84

Prepared by:

  
Stephen J. LaFleur

Approved for Public Release: Distribution Unlimited

Report Documentation Page			Form Approved OMB No. 0704-0188		
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE <b>05 DEC 1984</b>		2. REPORT TYPE <b>Technical Memo</b>		3. DATES COVERED <b>05-12-1984 to 05-12-1984</b>	
4. TITLE AND SUBTITLE <b>SIMAS ADM XBT Algorithm</b>			5a. CONTRACT NUMBER		
			5b. GRANT NUMBER		
			5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S) <b>Stephen LaFleur</b>			5d. PROJECT NUMBER <b>A90230</b>		
			5e. TASK NUMBER		
			5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <b>Naval Underwater Systems Center, New London Laboratory, New London, CT, 06320</b>			8. PERFORMING ORGANIZATION REPORT NUMBER <b>TM 841162</b>		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) <b>Naval Sea Systems Command</b>			10. SPONSOR/MONITOR'S ACRONYM(S) <b>NAVSEA</b>		
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)		
12. DISTRIBUTION/AVAILABILITY STATEMENT <b>Approved for public release; distribution unlimited</b>					
13. SUPPLEMENTARY NOTES <b>NUWC2015</b>					
14. ABSTRACT <b>An algorithm has been developed for the detection and correction of surface ship launched expendable bathythermograph (XBT) data that is manually implemented into the Sonar In-Situ Mode Assessment System (SIMAS). Reliability of the measured data is significantly improved over previous techniques used in SIMAS and, with slight modification, the algorithm can be adapted for use in totally automated surface ship performance prediction systems.</b>					
15. SUBJECT TERMS <b>XBT; SIMAS; expendable bathythermograph data; Sonar in-Situ Mode Assessment System</b>					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT <b>Same as Report (SAR)</b>	18. NUMBER OF PAGES <b>190</b>	19a. NAME OF RESPONSIBLE PERSON
a. REPORT <b>unclassified</b>	b. ABSTRACT <b>unclassified</b>	c. THIS PAGE <b>unclassified</b>			

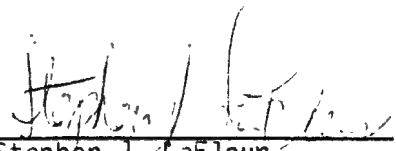
NAVAL UNDERWATER SYSTEMS CENTER  
NEW LONDON LABORATORY  
NEW LONDON, CONNECTICUT 06320

Technical Memorandum

SIMAS ADM XBT ALGORITHM

Date: 5 DEC 84

Prepared by:

  
Stephen J. LaFleur

Approved for Public Release: Distribution Unlimited

ABSTRACT

An algorithm has been developed for the detection and correction of surface ship launched expendable bathythermograph (XBT) data that is manually implemented into the Sonar In-Situ Mode Assessment System (SIMAS). Reliability of the measured data is significantly improved over previous techniques used in SIMAS and, with slight modification, the algorithm can be adapted for use in totally automated surface ship performance prediction systems.

ADMINISTRATIVE INFORMATION

This memorandum was produced under Job Order Number A90230. The principal investigator is George Brown, Code 3333. The sponsoring activity is Naval Sea Systems Command; Paul Tiedeman (NAVSEA 63D-3).

ACKNOWLEDGMENT

The author gratefully acknowledges the contribution of Eugene Podeszwa, Ronald Flight, Dr. Gustave Leibiger and George Brown who provided invaluable assistance throughout the development of this algorithm.



## INTRODUCTION

The expendable bathythermograph (XBT) has proven to be a valuable tool in providing a description of the ocean environment in the form of a temperature vs. depth trace. This XBT trace is the source of the most significant in-situ environmental data which is used by the Sonar In-Situ Mode Assessment System (SIMAS) and related system performance predictors being developed under the Acoustic Performance Prediction (APP) project to calculate such outputs as detection range predictions and propagation loss range functions. It is important to obtain as accurate a description of the ocean environment as is possible to achieve maximum reliability of the sonar performance prediction products. Due to the unreliability of the XBT device, it is necessary to compare the data from it with historical data (which is the result of many measurements and much analysis) in order to determine if the XBT data should be used at all. If it is determined that the XBT data is usable, it will be compared to the historical data again for any error correcting which may be necessary.

The purpose of this memo is to describe the techniques used in the XBT error correcting algorithm designed for a surface ship system which uses manually entered data. The algorithm has been translated to a FORTRAN 77 computer program and is presently being used in the version of the Sonar In-Situ Mode Assessment System (SIMAS) that is operational on the VAX 11/780 at NUSC/NL.

An algorithm has been designed for an automated system with the intended use being in submarine performance prediction systems (reference A). Using this algorithm as a guide and baseline, changes were made to allow the use of the same error correcting techniques in a manually operated system (i.e., a system where the data is entered at a terminal). Changes were also made so that the final product of the algorithm is not biased toward submarine performance prediction systems. In the case of the submarine automated system, depth-temperature pairs are provided by the XBT probe every four feet. This would result in 376 depth-temperature pairs if the XBT probe was accurate to a depth of 1500 feet. In the case of the manually operated surface ship system, the sonar operator must select the depth-temperature pairs from the XBT trace with the maximum number allowed to be entered set at 25. Typically, five to ten pairs are sufficient.

## DESCRIPTION OF TESTS

The following is a brief discussion of the tests used in the algorithm to determine if the XBT data should be accepted or rejected. The two tests performed on the XBT data are: (1) a realistic temperature range test, and (2) a test of deviation from the historical data (tolerance envelope test).

(1) The first test (temperature range test) is performed to ensure that the XBT data is within realistic temperature limits. The range to define the realistic bounds for the temperature is 27°F to 95°F. If more than half of the temperature values at depths shallower than 1500 feet are outside the allowable range, the data will be rejected. This test will, for example, detect a near surface wire break. A wire break or similar failure near the

surface cannot be corrected. If the XBT data fails this test, the operator is given the following choices:

1 = New BT

2 = Historical (See References B - F)

At this time it should be mentioned that the XBT temperature gradient extreme test and modification, which has been performed in previous versions of the algorithm, has been removed. There are three reasons for this. First, it was found that there was historical data which exceeded the limits being applied to the XBT temperature gradients. Second, it was found that modifying these gradients under certain situations could alter the layer depth of the XBT. Third, it was decided that altering the manually input XBT temperatures before comparing the XBT data against the historical tolerance envelope to determine acceptance or rejection of the XBT was undesirable. Adjustment of XBT points to the historical tolerance envelope, glitch removal and smoothing as described in "Description of the Algorithm" below will adjust any gradients which are truly extreme for the ocean area and month.

(2) The second test (tolerance envelope test) is a general examination for errors which compares the XBT data against the historical data's tolerance envelope. The calculation of the tolerance envelope is based on the fact that the water temperature varies within determinable limits about the historical data. Since the historical data is available in terms of sound speed, this test will use the XBT's sound speed values instead of temperature values. Studies show that the sound speed at the surface in a specific location varies less than  $\pm 15$  ft/sec about the mean and the variation tapers off so there is practically no variation at 2500 feet. The tolerance envelope is defined by the expression

$$SS \pm (15 - .006D)$$

where SS is the historical sound speed in ft/sec at depth D in feet. This expression defines the maximum allowable deviation in ft/sec from the historical value.

It was found that different operators were selecting different points from the XBT trace for manual entry into SIMAS. One operator might select most XBT points in a region where the XBT trace was within the tolerance envelope, while another operator might select most XBT points in a region where the XBT trace was outside of the tolerance envelope. Comparing this raw XBT data against the tolerance envelope could cause an acceptable XBT to be rejected or a bad XBT to be accepted, depending on which points the operator selected. In order to alleviate this problem and give equal weight to all the selected XBT data points, the XBT and historical sound speeds are now interpolated for every 4 feet of depth. One half the width of the historical tolerance envelope is also calculated for every 4 feet of depth. These temporary interpolated values are not used in the merge of the resultant profile and reside in two single variables as the algorithm loops from the surface to 1500 feet or the last XBT point (whichever comes first) with an increment of 4 feet. One counter (CNT) is incremented for every pass through the loop, and another counter (BAD) is incremented every time the XBT's interpolated sound

speed falls outside the tolerance envelope about the historical's interpolated sound speed. After leaving this loop (BAD) is compared to (CNT) and if more than half of the (CNT) points are (BAD), the XBT is rejected. If the XBT fails the tolerance envelope test the operator is given the following choices:

- 1 = New XBT or edit current XBT (Recommended)
- 2 = New SSP Area (Recommended)
- 3 = Use historical SSP (Recommended)
- 4 = Force BT to fit Historical (XBT data will be adjusted to use layer depth indicated by XBT and historical data below layer)
- 5 = Use XBT exactly as is (Not recommended because the XBT has been rejected and may produce unreliable results.)

#### DESCRIPTION OF THE ALGORITHM

Following is a description of the XBT algorithm which uses the 2 tests described above.

The XBT data can be entered at the terminal in metric or English temperatures or sound speeds. The algorithm automatically determines which form each data point was entered in, converts it if necessary, and fills one array with english depths, another array with english temperatures, and two arrays with english sound speeds. Leroy's equation is used to derive sound speed from temperature or temperature from sound speed.

The previous, current, and next months' historical Sound Speed Profiles (SSPs) are displayed graphically side by side (solid curves) with the XBT's depth-sound speed pairs (X's) overlaying each of them for operator comparison (see figure 1). The operator chooses the closest matching month which is to be used for merging with the XBT. This is especially useful if, for example, it is July first and the operator isn't sure whether to use June or July historical data for comparing the XBT data against. The operator is given the opportunity to get a hard copy of this graphics display.

Now the layer depths of the chosen month's historical SSP and the XBT's SSP are defined. This is done by checking the appropriate SSP to find the first sound speed value that is less than the one preceeding it. Once this sound speed value is found, the layer depth is defined as the depth corresponding to the sound speed value preceeding it.

Test (1) (Temperature Range Test) is performed at this time. (See 'Description of Tests' above)

Test (2) (Tolerance Envelope Test) is performed at this time. (See 'Description of Tests' above)

If the above two tests are passed, the XBT processing continues as described below.

It has been determined from the study described in reference A that the mean layer values for the North Pacific, North Atlantic, and Indian Oceans vary no greater than  $\pm 50$  feet 61 percent of the time for a specific location and month. Using this criteria of limiting the XBT layer depth to a range of

$\pm 50$  feet of the historical layer depth will not alter the final output of the performance prediction system significantly unless there is a shallow layer with a shallow source. In this case, the final predictions could be caused to be pessimistic or optimistic, depending on whether the layer was shifted to a shallower or deeper depth. The shifting of the layer to a shallower depth in the surface ship application will cause the final predictions to be pessimistic, i.e., the shallower the layer, the smaller the predicted range of coverage will be. To ensure that the most pessimistic outcome is always produced, the following limitation was put on the layer depth check. The XBT layer depth is compared with the historical layer depth. If the XBT layer depth  $\leq$  the historical layer depth and  $< 100$  feet, the XBT layer depth is used. If not, the XBT layer depth is again compared to the historical layer depth. If the XBT layer depth is within  $\pm 50$  feet of the historical layer depth, the XBT layer depth is used. If not, the XBT layer depth is modified to  $\pm 50$  feet of the historical layer depth.

The sound speeds which correspond to the manually entered XBT data points are now compared against the historical tolerance envelope. Working from the surface, any points which lie outside the envelope are moved to the edge of the envelope. If a point 2500 feet or deeper is encountered, the previous point becomes the last data point. This is done so that the XBT data can be merged smoothly into the historical data at 2500 feet where the tolerance envelope has a value of zero.

The data is now checked below the layer for glitches, or spurious changes in gradients that are not indicative of any actual environmental condition. It is impossible to have a glitch above the layer because the layer depth has been defined as the last point before the first negative gradient; therefore, the direction of the gradient above the layer could never change twice as described below. The glitch test compares the gradients between consecutive sound speed values. If the direction of the gradients between four consecutive data points changes twice, a glitch is present. The glitch is eliminated by removing the third data point of the four in question. This process is continued until all glitches are removed from the data.

After the glitches have been removed, the data is smoothed to eliminate the saw-toothed effect produced by limiting the data to the tolerance envelope. If there were no bad data points, i.e., all the data points fell inside the tolerance envelope, the data is not smoothed. A three-point smoothing routine is used which ensures that each data point is uniformly weighted. The layer depth point will not be modified by the smoothing routine.

After the manually entered XBT data has gone through the above checks and possible modifications, it is again compared against the historical tolerance envelope. Working from the surface, any points which lie outside the envelope are moved to the edge of the envelope. If a point 2500 feet or deeper is encountered, the previous point becomes the last data point. This is done so that the XBT data can be merged smoothly into the historical data at 2500 feet where the tolerance envelope has a value of zero. This is being done a second time because the layer depth check and modification could have added a point at 2500 feet or deeper. The final layer depth to be used by prediction routines is determined from the resultant merged profile produced in the next step.

Finally, the data is extended to 2500 feet where the historical deep profile can be appended to the XBT data to produce a continuous profile from the surface to the bottom of the ocean. The data is extended by using the following expression.

$$(2500 - DS(L))/((DEPDIF)(VELDIF) + VS(L))$$

where: DS(L) = Historical depth (in feet) at which sound speed is re-calculated.  
 DEPDIF = Depth difference (in feet) between the last XBT point and 2500 feet.  
 VELDIF = Sound speed difference (in feet per second) between the XBT data and historical data at the depth of the last XBT point.  
 VS(L) = Historical sound speed (in feet per second) at the depth of the point at which sound speed is re-calculated.

The resultant profile is now displayed graphically on the CRT along with an environmental data summary (see figure 2). The operator is given the opportunity to get a hard copy of this graphics display. A second environmental summary with more detail and no graphics is always output to the disk and the line printer (see figure 3).

A flowchart describing this algorithm is provided in APPENDIX A. APPENDIX B shows the logical calling sequence of the subroutines used in this algorithm. The FORTRAN 77 subroutines are listed alphabetically in APPENDIX C.

#### SUMMARY

Several new features have been incorporated into this version of the XBT processing algorithm which are very useful. This version allows the operator to use his judgement by graphically comparing the XBT's SSP to the historical SSPs of the previous, current and next month allowing him to choose the historical SSP which most closely matches the XBT's SSP for merging purposes. This could mean the difference between rejection and acceptance of the XBT especially if the current date is close to a month boundary.

The operator is also allowed to make the above comparisons with the historical SSP's of adjacent SSP areas (see figure 4). This allows the operator to move through space as well as time and it could also mean the difference between rejection and acceptance of the XBT especially if the platform is operating close to the boundary of a SSP area. These graphical decision aids show the operator how close the XBT data is to the historical data and give him some leeway in selecting the best historical data when boundary conditions exist.

The gradient extremes check and modification has been removed thereby eliminating several problems.

The tolerance envelope test has been modified to give equal weight to all XBT data points making the acceptance/rejection of the XBT more uniform between different operator entries for the same XBT trace. This acceptance/

TM No. 841162

rejection uniformity will be further improved when the surface ship XBT data input is automated thereby eliminating the need for operators to select points from an XBT trace for manual entry into the computer.

REFERENCES

- (A) G. A. Leibiger, E. M. Podeszwa, XBT Processing and Error Correcting Algorithm, NUSC TM 781154, of 3 August 1978
- (B) E. M. Podeszwa, Sound Speed Profiles for the North Pacific Ocean, NUSC TD 5271, of 2 February 1976 (2nd Printing - April 1981).
- (C) E. M. Podeszwa, Sound Speed Profiles for the North Atlantic Ocean, NUSC TD 5447, of 20 October 1976 (2nd Printing - April 1981).
- (D) E. M. Podeszwa, Sound Speed Profiles for the Mediterranean Sea, NUSC TD 6309, of 15 Aug 1980
- (E) E. M. Podeszwa, Sound Speed Profiles for the Indian Ocean, NUSC TD 5555, of 11 December 1976 (2nd Printing - April 1981).
- (F) E. M. Podeszwa, Sound Speed Profiles for the Norwegian Sea, NUSC TD 6035, of 4 June 1979 (2nd Printing - April 1981).
- (G) E. M. Podeszwa, A Validation Study of the SIMAS Environmental Data Base and XBT Processing, NUSC TD 811023, of 15 March 1981
- (H) E. M. Podeszwa, XBT Processing in TOPPS and SIMAS, NUSC TD 831159, of 15 December 1983

TM No. 841162

SIMAS ADM XBT ALGORITHM  
T. M. No. 841162  
Stephen J. LaFleur  
Surface Ship Sonar Department  
15 Sep 84  
J.O. A90230  
UNCLASSIFIED

DISTRIBUTION LIST

External with Appendix B and Appendix C

NAVSEA (SEA 63D3, P. Tiedeman)

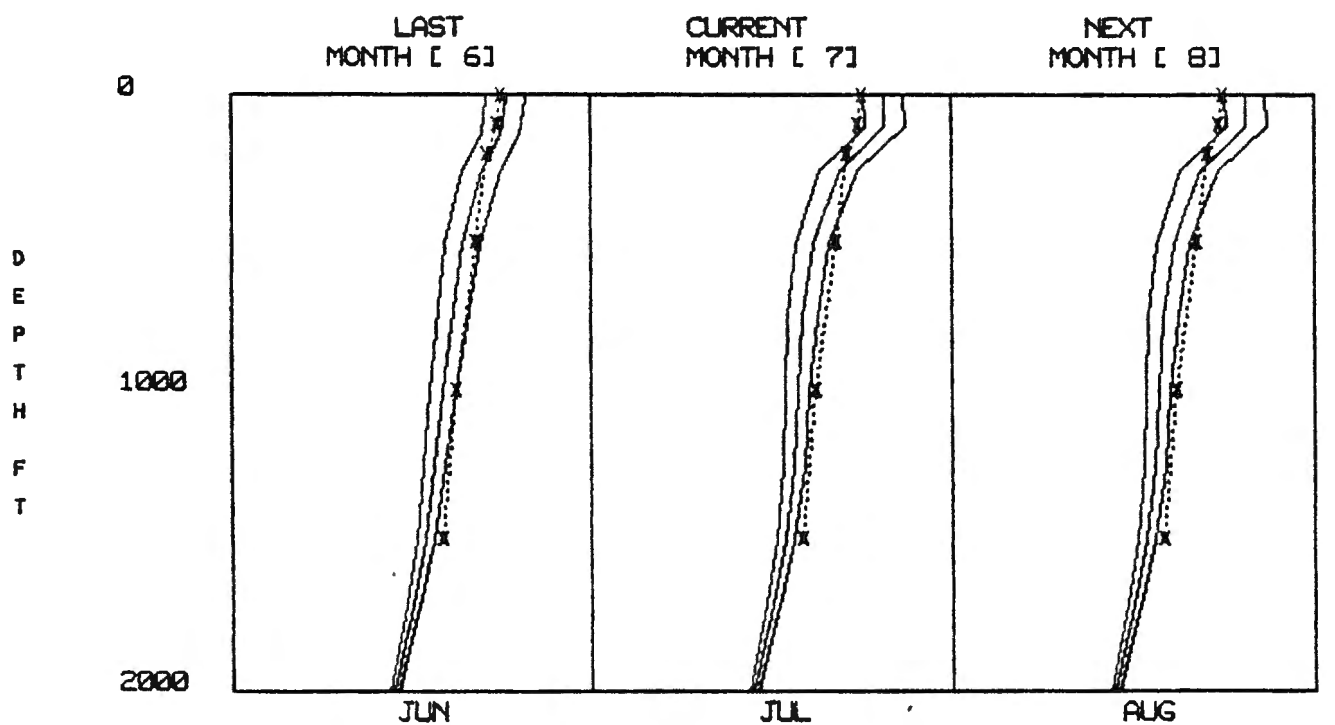
Internal without Appendix B and Appendix C

Code 0211	R. Bernier	Code 3331	P. Koenigs
021311	(3 cys) NLON Lab	3331	R. Dullea
021321	(3 cys) NPT Lab	3331	J. Chester
039	P. Genung	3331	J. Syck
3202	T. Fries	333	W. Schumacher
3203	G. Leibiger	3332	H. Weinberg
3202	C. Batts	3332	E. Jensen
3202	D. Yarger	3333	G. Brown
322	J. Geary	3333	M. Biggins
3224	D. McKinley	3333	R. Burnette
33	L. Freeman	3333	T. Cannan
33A	S. Santaniello	3333	S. Dasinger
33A	B. Cole	3333	E. Gannon
01Y	F. DiNapoli	3333	P. Griffin
33A	J. Gallagher	3333	R. Kenyon
33A2	CDR J. Hughes	3333	R. O'Conner
33A3	W. Roderick	3333	E. Podeszwa
33B1	A. Goodman	3333	J. Prentice
33B3	D. Counsellor	3333	J. Senkow
33B9	LCDR F. Welles	3333	W. Wachter
33C	J. Hanrahan	3333	F. Farmer
33C	D. Ashworth	3636	L. Mellberg
33C1	J. Bairstow	60	J. Keil
33C2	M. Gerber	601	T. Bell
331	R. Boivin	601	M. Pastore
3312	J. Bednarz	61	J. Cohen
3331	D. Browning	61	R. Almeida
3331	M. Tattersall	62	W. Pittsley
3331	W. Hauck	62	K. Korolenko
3331	J. Monti		
3331	R. Nielsen		

Internal with Appendix B and Appendix C

Code 33B1	W. Martin
3319	B. Thorp
3333	S. LaFleur (9)
601	H. J. Doeblor





NORTH ATLANTIC OCEAN

\*\*\* RAW BT DATA IS SHOWN WITH X'S \*\*\*

\*\*\* HISTORICAL SSP DATA IS SHOWN WITH SOLID LINES \*\*\*

SSP AREA: 31

ENTER MONTH# YOU THINK MATCHES THE XBT BEST

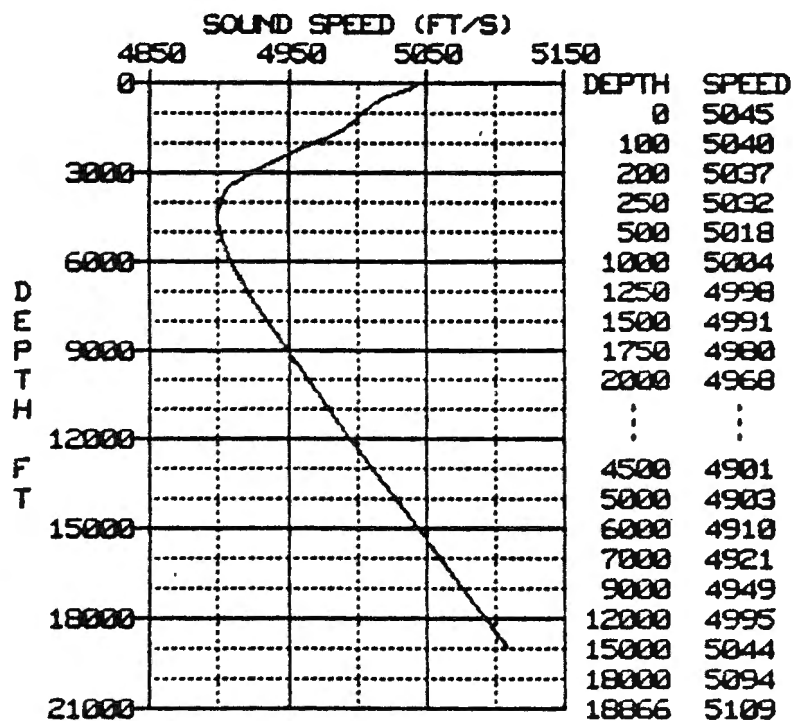
HOW MANY HARD COPIES WOULD YOU LIKE? [0,1,2,ETC.]

6

2

Figure (1)

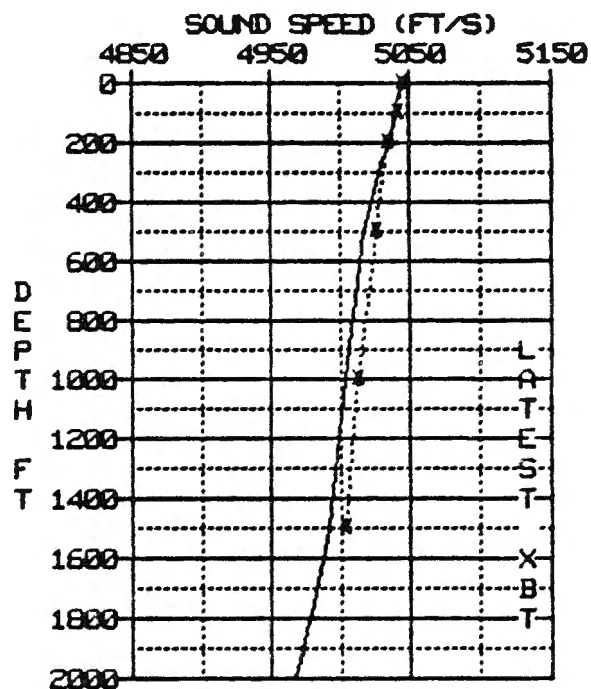
DATE AND TIME OF IN-SITU UPDATE IN FORECAST AREA JUL  
 NORTH ATLANTIC OCEAN AREA: 31 LAT: 1 0 0-N LONG: 1 0 0-E



TRUE BOTTOM DEPTH 3144 FATHOMS

TRUE WIND SPEED 11 KNOTS

\*\*\*\*\*CANDIDATE ACOUSTIC PATHS> - - - -



LAYER DEPTH 0 FT  
 SOUND CHNL. -NOT USABLE  
 CNVRG. ZONE-RANGE 72 KYDS  
 BTM. BOUNCE-MGS PROV. 3

\*\*\*\*\* RAW DATA IN SHALLOW SVP WITH X'S \*\*\*\*\*

HOW MANY HARD COPIES WOULD YOU LIKE? [0,1,2, ETC.] 1

Figure (2)

DATA TO BE USED FOR FORECASTING: JUL  
NORTH ATLANTIC OCEAN LAT: 1 0 0 N LONG: 1 0 0 E

SOUND VELOCITY PROFILE DATA  
LATEST XBT

NO.	DEPTH	VELOCITY
1	0.0	5045.3
2	100.0	5040.6
3	200.0	5037.9
4	250.0	5032.3
5	500.0	5018.2
6	1000.0	5004.2
7	1250.0	4998.1
8	1500.0	4991.1
9	1750.0	4980.1
10	2000.0	4968.1
11	2250.0	4955.0
12	2500.0	4944.0
13	2750.0	4933.0
14	3000.0	4924.0
15	3250.0	4915.0
16	3500.0	4909.0
17	4000.0	4902.0
18	4500.0	4901.0
19	5000.0	4903.0
20	6000.0	4910.0
21	7000.0	4921.0
22	9000.0	4949.0
23	12000.0	4995.0
24	15000.0	5044.0
25	18000.0	5094.0
26	18866.3	5109.0

SALINITY IS 36.50

MGS AREA IS 3

SSP AREA IS 31

CHART OR FATHOMETER BOTTOM DEPTH IS 3030 FATHOMS

CORRECTED BOTTOM DEPTH IS 3144 FATHOMS

TRUE WIND SPEED IS 11 KNOTS

\*\*\*CANDIDATE ACOUSTIC PATHS\*\*\*

LAYER DEPTH IS 0 FEET

CONVERGENCE ZONE RANGE IS 72.7 KYDS

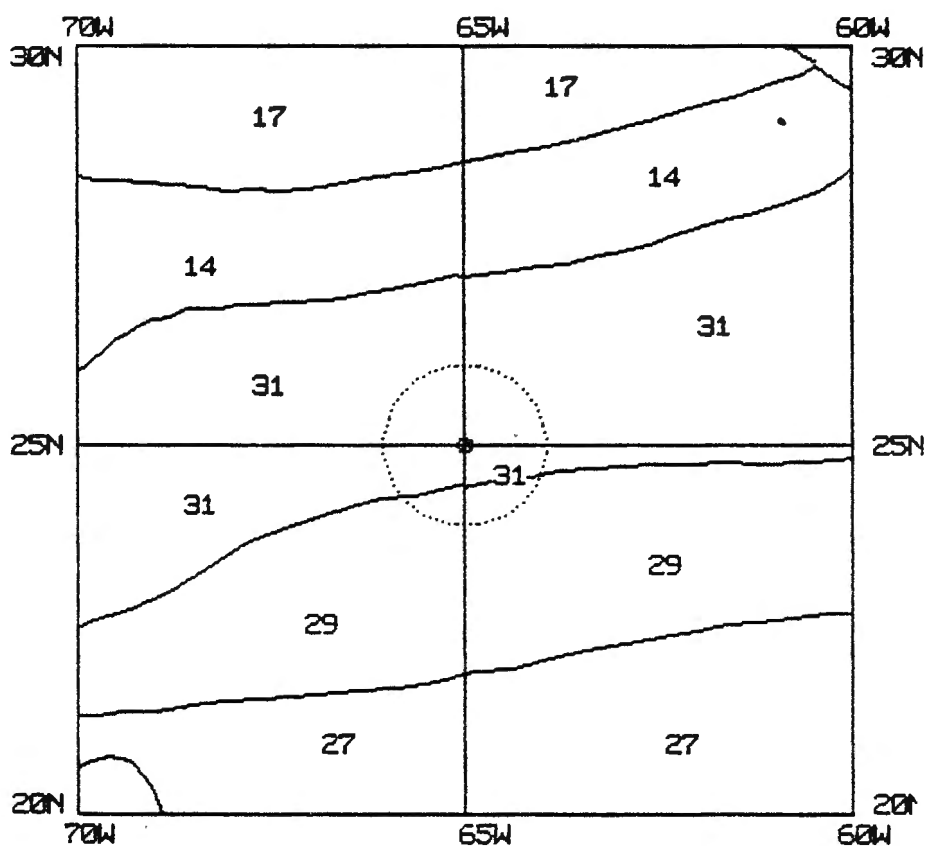
SOUND CHANNEL NOT USABLE

Figure (3)

# North Atlantic Ocean

60 NMI Range List  
(within dotted circle)

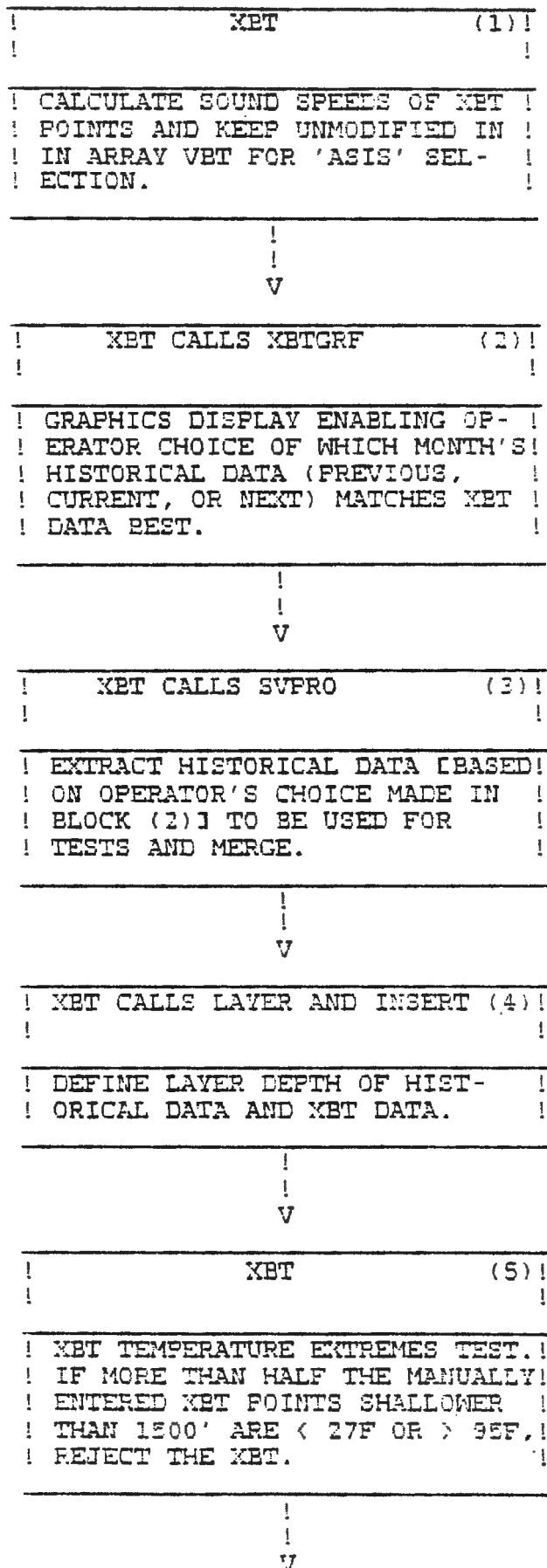
Code	Distance
31	Inter
29	30



THE SSP VALUE IS 31  
ENTER NEW VALUE (RETURN KEEPS CURRENT VALUE) 29

FIGURE 4

## FLOWCHART OF SIMAS ADM XBT ALGORITHM



## APPENDIX A

---

! XBT CALLS DUPDEP (6)!  
!

---

! ELIMINATE DUPLICATE CONSEC-!  
! UTIVE DEPTHS FROM XBT DATA. !

---

!  
!  
V

---

! XBT CALLS DUFVEL (7)!  
!

---

! ELIMINATE DUPLICATE CONSEC-!  
! UTIVE SOUND SPEEDS FROM XBT !  
! DATA. !

---

!  
!  
V

---

! XBT CALLS XBTCHK (8)!  
!

---

! TOLERANCE ENVELOPE TEST. !  
! INTERPOLATE HISTORICAL AND XBT !  
! SOUND SPEEDS FOR EVERY 4' OF !  
! DEPTH. IF MORE THAN HALF OF !  
! THE INTERPOLATED XBT SOUND !  
! SPEEDS SHALLOWER THAN 1500' !  
! ARE OUTSIDE OF THE TOLERANCE !  
! ENVELOPE ABOUT THE INTERPOLAT- !  
! ED HISTORICAL SOUND SPEEDS, !  
! REJECT THE XBT. !

---

!  
!  
V

---

! XBT CALLS LYRMOD (9)!  
!

---

! IF THE XBT LAYER DEPTH IS < !  
! 100' AND <= THE HISTORICAL !  
! LAYER DEPTH, USE THE XBT LAYER !  
! DEPTH. IF NOT, COMPARE LAYER !  
! DEPTHS AGAIN. IF THE XBT !  
! LAYER DEPTH IS WITHIN + OR - !  
! 50' OF THE HISTORICAL LAYER !  
! DEPTH, USE THE XBT LAYER DEPTH !  
! DEPTH. IF NOT, MODIFY THE !  
! XBT LAYER DEPTH TO + OR - 50' !  
! OF THE HISTORICAL LAYER DEPTH. !

---

!  
!  
V

---

! XBT CALLS XBTMOD (10)!  
!

---

! MOVE ANY MANUALLY ENTERED XBT !  
! POINTS WHICH FALL OUTSIDE THE !  
! HISTORICAL TOLERANCE ENVELOPE !  
! TO THE ENVELOPE. REMOVE THE !  
! LAST CONSECUTIVE GROUP OF !  
! POINTS WHICH ARE OUTSIDE THE !  
! TOLERANCE ENVELOPE. !

---

!  
!  
V

---

! XBT CALLS GLITCH (11)!  
!

---

! IF THERE ARE AT LEAST 3 POINTS!  
! DEEPER THAN THE XBT LAYER !  
! DEPTH, PERFORM THE GLITCH TEST!  
! AND REMOVE ANY GLITCHES BELOW !  
! THE LAYER. !

---

!  
!  
V

---

! XBT CALLS DUPDEP (12)!  
!

---

! ELIMINATE DUPLICATE CONSEC- !  
! UTIVE DEPTHS FROM XBT DATA. !

---

!  
!  
V

---

! XBT CALLS DUPVEL (13)!  
!

---

! ELIMINATE DUPLICATE CONSEC- !  
! UTIVE SOUND SPEEDS FROM THE !  
! XBT DATA. !

---

!  
!  
V

---

! XBT CALLS LAYER AND INSERT (14)!  
!

---

! DEFINE LAYER DEPTH OF XBT DATA!

---

!  
!  
V

---

```
! XBT CALLS SMOOTH (15)!
!
```

---

```
! IF ANY OF THE INTERPOLATED XBT!
! SOUND SPEEDS WERE OUTSIDE THE !
! TOLERANCE ENVELOPE ABOUT THE !
! INTERPOLATED HISTORICAL SOUND !
! SPEEDS [IN BLOCK (8)], SMOOTH !
! ALL THE MANUALLY ENTERED AND !
! CORRECTED XBT DATA POINTS !
! EXCEPT THE LAYER DEPTH POINT. !
```

---

```
!
!
V
```

---

```
! XBT CALLS XBTMOD (16)!
!
```

---

```
! DO BLOCK (9) AND EXTEND THE !
! RESULTING XBT DATA TO 2500'. !
```

---

```
!
!
V
```

---

```
! AFTER RETURNING TO (17)!
! ENVIRN OR FORCET FROM XBT !
```

---

```
! EXTEND THE XBT DATA (RESULTING!
! FROM BLOCK (16)) WITH THE !
! DEEP HISTORICAL DATA TO FORM !
! FORM THE FINAL RESULTANT SOUND!
! SPEED PROFILE FROM THE OCEAN'S!
! SURFACE TO THE OCEAN'S BOTTOM !
! FOR USE BY PREDICTION ROUTINES!
! .DEFINE THE LAYER DEPTH OF THE!
! RESULTANT PROFILE. !
```

---



APPENDIX C CAN BE OBTAINED FROM THE AUTHOR UPON REQUEST. Call:

AUTOVON 636-440-4337  
COMMERCIAL (203)-440-4337

## APPENDIX B

The following source code modules are shown as they logically appear in the current version of the SIMAS ADM XBT ALGORITHM. These modules are all written in FORTRAN 77 and have each been documented with prologue, block comments, and line comments in accordance with the Acoustic Performance Prediction Software Architecture Plan of 1 Nov 1981, revised 9 Dec 1982.

SIMAS

APPENDIX B

ENVIRN  
MAP \*\*  
BT  
EDITBT  
METRIC  
LEROY  
VELTMP  
LEROY  
VELTMP  
LEROY  
INSERT  
LAYER  
NOCONV  
INSERT  
ACOUS  
SVPGRF  
INSERT  
CONECT  
DRAW  
CLIPPOS  
CLIPT  
INITT  
INUMBR  
TEXT  
LNTYPE  
MOVE  
OUTPUT  
TEXT  
XBT \*  
FORCST  
MAP \*\*  
KEYPCH  
BT  
KSCAT  
LATLNG  
SVPRO  
SSP  
VELTMP  
LEROY  
INSERT  
LAYER  
NOCONV  
INSERT  
ACOUS  
SVPGRF  
INSERT  
CONECT  
DRAW  
CLIPPOS  
CLIPT  
INITT  
INUMBR  
TEXT  
LNTYPE  
MOVE  
OUTPUT  
TEXT  
XBT \*

\* XBT  
  LEROY  
  XBTGRF  
  SVPRO  
  LAYER  
  INSERT  
  XBTERR  
    ASIS  
      INSERT  
      METRIC  
  DUPDEP  
  DUPVEL  
  XBTCHK  
  XBTMOD  
    ASIS  
      INSERT  
  LYRMOD  
  GLITCH  
  SMOOTH

```
** MAP
  SETOAC
    FSETUP
      OPNFIL
    SETPOS
      FLOOR
    INDX
  CRUNCH
    OPNFIL
    INDX
    GETREC
      BMOD
    CNNCT
      KMOD
    END1
      KMOD
      BMOD
      FNP
    END2
      KMOD
      BMOD
      FNP
  DNUT
    KMOD
    END1
      KMOD
      BMOD
      FNP
    END2
      KMOD
      BMOD
      FNP
    FNP
  PDIST
    END1
      KMOD
      BMOD
      FNP
    END2
      KMOD
      BMOD
      FNP
    BMOD
  GRAPH
    BMOD
    FNP
```

## APPENDIX C

ACOUS  
ASIS  
BMOD  
BT  
CLIPPOS  
CLIPT  
CNNCT  
CONNECT  
CRUNCH  
DNUT  
DRAW  
DUPDEP  
DUPVEL  
EDITBT  
END1  
END2  
ENVIRN  
FLOOR  
FNP  
FORCST  
FSETUP  
GETREC  
GLITCH  
GRAPH  
INDX  
INITT  
INSERT  
INUMBR  
KEYPCH  
KMOD  
KSCAT  
LATLNG  
LAYER  
LEROY  
LNTYPE  
LYRMOD  
MAP  
METRIC  
MOVE  
NOCONV  
OPNFIL  
OUTPUT  
PDIST  
SETOAC  
SIMAS  
SMOOTH  
SSP  
SVPGRF  
SVPRO  
TEXT  
VELTMP  
XBT  
XBTCHK  
XBTErr  
XBTGRF  
XBTMOD

```

0001      SUBROUTINE ACOUS(Z,V,NPT,CV,T,H,G,S,TP)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: ACOUS
0005      ! AUTHOR: G. BROWN & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1974 & 12/83 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE ACOUS USES INPUT OF SOUND SPEED PROFILE AND
0008      !              VERTEXING VELOCITY TO COMPUTE TWO-WAY TRAVEL TIME,
0009      !              HORIZONTAL RANGE, SPREADING LOSS, SLANT RANGE, AND TIME
0010      !              VELOCITY GRADIENT.
0011      ! INPUTS:  PARAMETERS PASSED IN.
0012      ! OUTPUTS: PARAMETERS PASSED OUT.
0013      ! MODULES CALLED: NONE
0014      ! CALLED BY: ACTV26,BBTBLS,OTHERS,NOCONV
0015      !
0016      ! VARBL  SIZE      PURPOSE                                TYPE      RANGE
0017      ! -----
0018      ! CV      VERTEX VELOCITY                                REAL*4
0019      ! DG      FACTOR                                          REAL*4
0020      ! DZ      DEPTH FACTOR                                    REAL*4
0021      ! G      SPREADING LOSS                                  REAL*4
0022      ! GJ      GRADIENT:SS WITH RESPECT TO DZ                REAL*4
0023      ! H      HORIZ RANGE FOR CURRENT SPEED                  REAL*4
0024      ! HJ      FACTOR                                          REAL*4
0025      ! J      COUNTER                                          INTEGER*2
0026      ! JUP     NUMBER OF DATA POINTS MINUS 1                INTEGER*2
0027      ! NPT     NUMBER OF DATA POINTS                        INTEGER*2
0028      ! R1      FACTOR                                          REAL*4
0029      ! R1R2    FACTOR                                          REAL*4
0030      ! R2      FACTOR                                          REAL*4
0031      ! S      SLANT RANGE                                      REAL*4
0032      ! T      TWO-WAY TRAVEL TIME                            REAL*4
0033      ! TP     TIME-VELOCITY GRADIENT                        REAL*4
0034      ! V      (1)  VELOCITY OF DEPTH/VEL ARRAY              REAL*4
0035      ! VJ2     FACTOR                                          REAL*4
0036      ! Z      (1)  DEPTH OF DEPTH/VEL ARRAY                 REAL*4
0037      !
0038      INTEGER*2 J,JUP,NPT
0039      REAL*4    CV,DG,DZ,G,GJ,H,HJ,R1,R1R2,R2,S,T,TP,V,VJ2,Z
0040      DIMENSION Z(1),V(1)
0041
0042      !-----PRELIMINARIES-----
0043      JUP=NPT-1      ! NUMBER OF DATA POINTS - 1
0044      T=0.           ! INITIALIZE 2-WAY TRAVEL TIME
0045      TP=0.0         ! INITIALIZE TIME/VEL PROFILE
0046      H=0.           ! INITIALIZE HORIZ RANGE
0047      G=0.           ! INITIALIZE SPREADING LOSS
0048      S=0.           ! INITIALIZE SLANT RANGE
0049      R1=(CV-V(1))*(CV+V(1)) ! FACTOR
0050      IF (R1.LT.0.) R1=0. ! DISALLOW NEGATIVE VALUE
0051      R1=SQRT(R1)    ! FACTOR
0052
0053      !-----INTERMEDIATE VALUES-----
0054      DO 100 J=1,JUP ! DO UNTIL # OF PTS - 1
0055          R2=(CV-V(J+1))*(CV+V(J+1)) ! FACTOR
0056          IF (R2.LT.0.) R2=0.         ! DISALLOW NEGATIVE VALUE
0057          R2=SQRT(R2)                 ! FACTOR
0058          DZ=Z(J+1)-Z(J)              ! DEPTH FACTOR
0059          GJ=(V(J+1)-V(J))/DZ         ! GRADIENT:SS WITH RESPECT TO DZ

```

```

0060      HJ=1./GJ*(R1-R2)          ! FACTOR
0061      H=H+HJ                    ! HORIZ RANGE FOR CURRENT SPEED
0062      VJ2=1.                     ! FACTOR
0063      IF (R2.GT.0.) VJ2=V(J+1)/(CV+R2) ! FACTOR
0064      T=T+1./GJ*ALOG((CV+R1)/V(J)*VJ2) ! TWO-WAY TRAVEL TIME
0065      R1R2=R1*R2                  ! FACTOR
0066      R1=R2                       ! FACTOR
0067      IF (R1R2.LE.0.) GO TO 200    ! GO TO EQUATIONS AND EXIT
0068      DG=HJ/(R1R2)                ! FACTOR
0069      G=G+DG                      ! SPREADING LOSS
0070      TP=TP-DG                    ! TIME-VELOCITY GRADIENT
0071      S=S+CV/GJ*(ACOS(V(J)/CV)-ACOS(V(J+1)/CV)) ! SLANT RANGE
0072      100      CONTINUE           ! END DO LOOP
0073
0074      !-----FINAL VALUES-----
0075      200      T=4.*T              ! TWO-WAY TRAVEL TIME
0076              TP=4.*TP            ! TIME-VELOCITY GRADIENT
0077              H=2.*H/3.           ! HORIZ RANGE; SPREADING LOSS
0078              G=10.0*ALOG10(H*2./3.*G*(CV/V(1))**2*(CV**2-V(1)**2))
0079              S=2.*S/3.0          ! SLANT RANGE
0080
0081      RETURN                      ! RETURN TO CALLING ROUTINE
0082      END                        ! END SUBROUTINE

```

#### COMMAND QUALIFIERS

```

FORTRAN /CHECK=ALL/LIST/SHOW=(INCLUDE,NOMAP) DBA3:[LAFLEUR]ACOUS.F77;1

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE_FORM)
/SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)
/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD_LINES /NOCROSS_REFERENCE

```

#### COMPILATION STATISTICS

```

Run Time:          1.63 seconds
Elapsed Time:      4.86 seconds
Page Faults:       348
Dynamic Memory:    117 pages

```



```

0001          SUBROUTINE ASIS(NUMBER,DEPTH,SPEED)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: ASIS
0005      ! AUTHOR: S. LAFLEUR & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1982 & 12/83 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE ASIS ALLOWS THE OPERATOR USE BT DATA
0008      !              AS IS.
0009      ! INPUTS: HARD COPY SELECTION, OPERATOR SELECTION TO UPDATE
0010      !              PARAMETERS OR NOT. VARIABLES IN COMMONS.
0011      ! OUTPUTS: CRT PROMPTING MESSAGES TO OPERATOR
0012      ! MODULES CALLED: INSERT
0013      ! CALLED BY: XBTMOD, XBTErr
0014      !
0015          INCLUDE 'DTV.INC'
0016      1 ! -----DTV-----
0017      1 !  VARBL  SIZE      PURPOSE                                TYPE      RANGE
0018      1 !  -----
0019      1 !  D      (25)     DEPTH                                REAL*4
0020      1 !  DD     (25)     DEPTH                                REAL*4
0021      1 !  NNBT                                NUMBER OF BATHETHERMAL    INTEGER*2
0022      1 !  T      (25)     TEMPERATURE                        REAL*4
0023      1 !  TT     (25)     TEMPERATURE                        REAL*4
0024      1 !  VEL    (25)     VELOCITY                            REAL*4
0025      1 !
0026      1          INTEGER*2 NNBT
0027      1          REAL*4 D,DD,T,TT,VEL
0028      1
0029      1          COMMON /DTV/  D(25),T(25),VEL(25),DD(25),TT(25),NNBT
0030      1 ! -----END DTV-----
0031          INCLUDE 'ENVN.INC'
0032      1 ! -----ENVN-----
0033      1 !  VARBL  SIZE      PURPOSE                                TYPE      RANGE
0034      1 !  -----
0035      1 !  BIO     (2)      BIOLOGICAL BACK SCATTERING    REAL*4      -57. & -47.
0036      1 !  DLYR                                LAYER DEPTH        REAL*4
0037      1 !  MGS                                MGS PROVINCE        INTEGER*2
0038      1
0039      1          REAL*4 BIO,DLYR
0040      1          INTEGER*2 MGS
0041      1          DATA BIO/-57.,-47./
0042      1
0043      1          COMMON /ENVN/ BIO(2),DLYR,MGS
0044      1
0045      1 ! -----END ENVN-----
0046          INCLUDE 'SVP.INC'
0047      1 ! -----SVP-----
0048      1 !  VARBL  SIZE      PURPOSE                                TYPE      RANGE
0049      1 !  -----
0050      1 !  BDF                                BOTTOM DEPTH IN FATHOMS    REAL*4
0051      1 !  BIOP                                BIOLOGICAL BACK SCATTERING COEF    REAL*4
0052      1 !  BTDATE (9)      DATE OF LAST BT INPUT        BYTE
0053      1 !  BTTIME (8)      TIME OF LAST BT INPUT        BYTE
0054      1 !  C      (50)      VELOCITY (PAIRED WITH Z FOR SVP)    REAL*4
0055      1 !  CC     (50)      VELOCITY (PAIRED WITH ZZ FOR SVP)    REAL*4
0056      1 !  CS                                SOUND VELOCITY AT SURFACE    REAL*4
0057      1 !  DEG                                TEMPERATURE (DEG)        REAL*4      57.2957795
0058      1 !  EL                                LAYER DEPTH            DATA
0059      1 !  F      FREQUENCY                            REAL*4

```

```

0060 1 ! GRDS          GRIDS          REAL*4          0.0164
0061 1 ! ITO          MINIMAL 2-WAY TRAVEL TIME    INTEGER*2
0062 1 ! MGSOP        MGS PROVINCE NUMBER    INTEGER*2
0063 1 ! N            # OF DEPTH/VELOCITY PAIRS    INTEGER*2
0064 1 ! NN          # OF DEPTH/VELOCITY PAIRS    INTEGER*2
0065 1 ! PI          MATHEMATICAL CONSTANT PI      REAL*4          3.1415927
0066 1 ! SNDATE (9)   DATE SYS PARMS LAST UPDATED  BYTE
0067 1 ! SNTIME (8)   TIME SYS PARMS LAST UPDTAED  BYTE
0068 1 ! SYDATE (9)   CURRENT DATE READ FROM SYSTEM BYTE
0069 1 ! SYTIME (8)   CURRENT TIME READ FROM SYSTEM BYTE
0070 1 ! TMP          TEMPERATURE                REAL*4
0071 1 ! UMKZ        BOTTOM BACK SCATTERING COEF.    REAL*4          -28.0
0072 1 ! WS          WIND SPEED                  REAL*4
0073 1 ! Z            (50) DEPTH OF POINT OF SOUND SPEED REAL*4
0074 1 ! ZZ          (50) DEPTH OF POINT OF SOUND SPEED REAL*4
0075 1
0076 1              INTEGER*2 ITO,MGSOP,N,NN
0077 1              REAL*4   BDF,BIOP,C(50),CC(50),CS,DEG,EL,F,GRDS
0078 1              REAL*4   PI,TMP,UMKZ,WS,Z(50),ZZ(50)
0079 1              BYTE     SYDATE(9),SYTIME(8),BTDATE(9),BTTIME(8)
0080 1              BYTE     SNDATE(9),SNTIME(8)
0081 1              DATA    PI,DEG,GRDS/3.1415927,57.2957795,0.0164/
0082 1              DATA    UMKZ/-28./
0083 1
0084 1              COMMON /SVP/ F,N,Z,C,EL,MGSOP,BDF,WS,CS,TMP,BIOP,
0085 1              1        UMKZ,PI,DEG,GRDS,ITO,ZZ,CC,NN,
0086 1              2        SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME
0087 1 ! -----SVP-END-----
0088
0089 1              INCLUDE 'SVP1.INC'
0090 1 ! -----SVP1-----
0091 1 ! VARBL  SIZE  PURPOSE          TYPE      RANGE
0092 1 ! -----
0093 1 ! BUFFER (224) HISTORICAL DATA FILE BUFFER    REAL*4
0094 1 ! DS      (30) HISTORICAL DEPTH                REAL*4
0095 1 ! J20          # OF DEEP OCEAN DEPTH/VEL PAIRS    INTEGER*2
0096 1 ! NS          TOTAL # OF PAIRS IN HISTORICAL    INTEGER*2
0097 1 ! NSN        MONTH NUMBER (1=JAN.,ETC)          INTEGER*2    1 TO 12
0098 1 ! SLNTY      SALINITY                          REAL*4
0099 1 ! VS          (30) HISTORICAL VELOCITY            REAL*4
0100 1              REAL*4   BUFFER,DS,SLNTY,VS
0101 1              INTEGER*2 J20,NSN,NS
0102 1
0103 1              COMMON /SVP1/ J20,BUFFER(224),NSN,SLNTY,DS(30),VS(30),NS
0104 1 ! -----END SVP1-----
0105 1
0106 1 ! VARBL  SIZE  PURPOSE          TYPE      RANGE
0107 1 ! -----
0108 1 ! DEPDIF      DIFFERNECE IN DEPTH                REAL*4
0109 1 ! DEPTH      (25) DEPTH                          REAL*4
0110 1 ! ICNT        COUNTER                            INTEGER*2
0111 1 ! INDEXH      INDEX OF HISTORICAL POINT            INTEGER*2
0112 1 ! J          COUNTER                            INTEGER*2
0113 1 ! L          COUNTER                            INTEGER*2
0114 1 ! M          COUNTER                            INTEGER*2
0115 1 ! NU          SVP INDEX                          INTEGER*2
0116 1 ! NUMBER      NUMBER OF BT POINTS                INTEGER*2
0117 1 ! SPEED      (25) SVP INDEX                      REAL*4
0118 1 ! VELDIF      DIFFERENCE IN VELOCITY            REAL*4

```

```

0119      !
0120      ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMON ***
0121
0122      INTEGER*2 ICNT,INDEXH,J,L,M,NU,NUMBER
0123      REAL*4     DEPDIF,DEPTH,SPEED,VELDIF
0124      DIMENSION DEPTH(25),SPEED(25)
0125
0126      !-----GET AND STORE EXISTING DATA "AS
0127      CALL INSERT(NS,DS,VS,DEPTH(NUMBER),NU) ! GET SVP FOR HISTORICAL
0128      ICNT=0                                ! INITIALIZE ICNT
0129      DO 100 J=1,NUMBER                     ! PUT BT INTO PROFILE "AS IS"
0130          ICNT=ICNT+1                       ! INCREASE COUNT
0131          Z(ICNT)=DEPTH(J)                  ! STORE DEPTH
0132          C(ICNT)=SPEED(J)                  ! STORE VELOCITY
0133      100  CONTINUE                         ! CONTINUE
0134          DEPDIF = 2500.-DEPTH(NUMBER)      ! DIFFERENCE IN DEPTH
0135          VELDIF=SPEED(NUMBER)-VS(NU)       ! DIFFERENCE IN SOUND SPEED
0136
0137      !-----FIND INDEX OF HIST PT DEEPER THAN LAST
0138      DO 200 INDEXH=1,NS                    ! DO FOR NUMBER OF HIST VALUES
0139          IF(DS(INDEXH).GT.DEPTH(NUMBER)+5.) GOTO 250 ! EXIT LOOP
0140      200  CONTINUE                         ! END DO LOOP
0141          INDEXH=NS                         ! SET HISTORICAL INDEX
0142      250  DO 300 L=INDEXH,NS               ! ADJUST HIST PTS FROM LAST
0143          IF(DS(L).GT.2500.) GOTO 350      ! BT PT TO 2500' TO FIT BT
0144          ICNT=ICNT+1                       ! INCREASE COUNT
0145          Z(ICNT)=DS(L)                     ! STORE DEPTH
0146          C(ICNT)=(2500-DS(L))/DEPDIF*VELDIF+VS(L)! STORE VELOCITY
0147      300  CONTINUE                         ! END DO LOOP
0148          GO TO 500                         ! RESET TO AVOID LOOP 400
0149
0150      !-----PUT HIST BELOW 2500' INTO RESULTANT PROFILE
0151      350  DO 400 M=L,NS                    ! DO FOR ALL HIST VALUES
0152          ICNT=ICNT+1                       ! INCREMENT COUNTER
0153          Z(ICNT)=DS(M)                     ! STORE DEPTH AS IS
0154          C(ICNT)=VS(M)                     ! STORE VELOCITY AS IS
0155      400  CONTINUE                         ! END DO LOOP
0156      500  N=ICNT                           ! SET NUMBER OF BT POINTS
0157          RETURN                             ! RETURN TO CALLING ROUTINE
0158          END                               ! END SUBROUTINE

```

```

0001      REAL*4 FUNCTION BMOD(A,B)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: BMOD
0005      ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006      ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007      ! DATE: 1982 & 6/84 (FORTRAN 77)
0008      ! FUNCTION: THIS FUNCTION IS DESIGNED TO CALCULATE THE CLOCK
0009      !               ARITHMETIC MODULO VALUE FROM THE TWO PARAMETERS PASSED.
0010      ! INPUTS: TWO PARAMETERS PASSED IN
0011      ! OUTPUTS: CLOCK ARITHMETIC MODULO VALUE
0012      ! MODULES CALLED: NONE
0013      ! CALLED BY: CRUNCH, END1, END2, GETREC, GRAPH
0014      !
0015      ! VARBL   SIZE      PURPOSE                                TYPE      RANGE
0016      ! -----
0017      ! A      THE MODULO DIVISOR (REAL*4)
0018      ! B      THE CLOCK ARITHMETIC MODULUS (INTEGER*2)
0019      !
0020      REAL*4 A
0021      INTEGER*2 B
0022
0023      BMOD=AMOD(A,FLOATI(B))      ! GET REMAINDER FROM A/B
0024      IF (A*B.LT.0.) BMOD=B+BMOD  ! IF A OR B <0, ADD MODULUS
0025      IF (B.EQ.0) BMOD=0.0        ! IF DIVISION BY 0, SET TO 0
0026
0027      RETURN                      ! RETURN TO CALLING ROUTINE
0028      END                        ! END SUBROUTINE

```

```

0001          SUBROUTINE BT(INSSP,NBT)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: BT
0005      ! AUTHOR: G. BROWN & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1974 & 12/83 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE BT IS USED FOR MANUAL ENTRY OF BT DATA
0008      !              (DEPTH AND TEMPERATURE VALUES).
0009      ! INPUTS:  OPERATOR INPUT OF DATA.  VARIABLES IN COMMONS.
0010      ! OUTPUTS: CRT PROMPTING MESSAGES TO OPERATOR.
0011      ! MODULES CALLED: EDITBT,METRIC
0012      ! CALLED BY: ENVIRN
0013      !
0014          INCLUDE 'DTV.INC'
0015      1 ! -----DTV-----
0016      1 !  VARBL   SIZE   PURPOSE                                TYPE   RANGE
0017      1 !  -----
0018      1 !  D       (25)   DEPTH                                REAL*4
0019      1 !  DD      (25)   DEPTH                                REAL*4
0020      1 !  NNBT                                NUMBER OF BATHETHERMAL  INTEGER*2
0021      1 !  T       (25)   TEMPERATURE                        REAL*4
0022      1 !  TT      (25)   TEMPERATURE                        REAL*4
0023      1 !  VEL     (25)   VELOCITY                           REAL*4
0024      1 !
0025      1          INTEGER*2 NNBT
0026      1          REAL*4 D,DD,T,TT,VEL
0027      1
0028      1          COMMON /DTV/  D(25),T(25),VEL(25),DD(25),TT(25),NNBT
0029      1 ! -----END DTV-----
0030      )030      INCLUDE 'SVP.INC'
0031      1 ! -----SVP-----
0032      1 !  VARBL   SIZE   PURPOSE                                TYPE   RANGE
0033      1 !  -----
0034      1 !  BDF                                BOTTOM DEPTH IN FATHOMS  REAL*4
0035      1 !  BIOP                                BIOLOGICAL BACK SCATTERING COEF  REAL*4
0036      1 !  BTDATE (9)   DATE OF LAST BT INPUT                BYTE
0037      1 !  BTTIME (8)   TIME OF LAST BT INPUT                BYTE
0038      1 !  C       (50)   VELOCITY (PAIRED WITH Z FOR SVP)  REAL*4
0039      1 !  CC      (50)   VELOCITY (PAIRED WITH ZZ FOR SVP) REAL*4
0040      1 !  CS                                SOUND VELOCITY AT SURFACE  REAL*4
0041      1 !  DEG                                TEMPERATURE (DEG)      REAL*4      57.2957795
0042      1 !  EL                                LAYER DEPTH        DATA
0043      1 !  F       (50)   FREQUENCY                          REAL*4
0044      1 !  GRDS                                GRIDS                REAL*4      0.0164
0045      1 !  ITO                                MINIMAL 2-WAY TRAVEL TIME  INTEGER*2
0046      1 !  MGSOP                                MGS PROVINCE NUMBER      INTEGER*2
0047      1 !  N       (50)   # OF DEPTH/VELOCITY PAIRS          INTEGER*2
0048      1 !  NN                                # OF DEPTH/VELOCITY PAIRS  INTEGER*2
0049      1 !  PI                                MATHEMATICAL CONSTANT PI  REAL*4      3.1415927
0050      1 !  SNDATE (9)   DATE SYS PARMS LAST UPDATED        BYTE
0051      1 !  SNTIME (8)   TIME SYS PARMS LAST UPDTAED        BYTE
0052      1 !  SYDATE (9)   CURRENT DATE READ FROM SYSTEM      BYTE
0053      1 !  SYTIME (8)   CURRENT TIME READ FROM SYSTEM      BYTE
0054      1 !  TMP                                TEMPERATURE          REAL*4
0055      1 !  UMKZ                                BOTTOM BACK SCATTERING COEF.  REAL*4      -28.0
0056      1 !  WS       (50)   WIND SPEED                        REAL*4
0057      1 !  Z       (50)   DEPTH OF POINT OF SOUND SPEED     REAL*4
0058      1 !  ZZ      (50)   DEPTH OF POINT OF SOUND SPEED     REAL*4
0059      1

```

```

0060 1      INTEGER*2 ITO,MGSOP,N,NN
0061 1      REAL*4    BDF,BIOP,C(50),CC(50),CS,DEG,EL,F,GRDS
0062 1      REAL*4    PI,TMP,UMKZ,WS,Z(50),ZZ(50)
0063 1      BYTE     SYDATE(9),SYTIME(8),BTDATE(9),BTTIME(8)
0064 1      BYTE     SNDATE(9),SNTIME(8)
0065 1      DATA    PI,DEG,GRDS/3.1415927,57.2957795,0.0164/
0066 1      DATA    UMKZ/-28./
0067 1
0068 1      COMMON /SVP/ F,N,Z,C,EL,MGSOP,BDF,WS,CS,TMP,BIOP,
0069 1          1      UMKZ,PI,DEG,GRDS,ITO,ZZ,CC,NN,
0070 1          2      SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME
0071 1 ! -----SVP-END-----
0072 1      INCLUDE 'SVP1.INC'
0073 1 ! -----SVP1-----
0074 1 !  VARBL  SIZE      PURPOSE                                TYPE      RANGE
0075 1 !  -----
0076 1 !  BUFFER (224)    HISTORICAL DATA FILE BUFFER          REAL*4
0077 1 !  DS      (30)    HISTORICAL DEPTH                      REAL*4
0078 1 !  J20          # OF DEEP OCEAN DEPTH/VEL PAIRS          INTEGER*2
0079 1 !  NS          TOTAL # OF PAIRS IN HISTORICAL            INTEGER*2
0080 1 !  NSN         MONTH NUMBER (1=JAN.,ETC)                 INTEGER*2  1 TO 12
0081 1 !  SLNTY       SALINITY                                  REAL*4
0082 1 !  VS      (30)    HISTORICAL VELOCITY                  REAL*4
0083 1
0084 1      REAL*4    BUFFER,DS,SLNTY,VS
0085 1      INTEGER*2 J20,NSN,NS
0086 1
0087 1      COMMON /SVP1/ J20,BUFFER(224),NSN,SLNTY,DS(30),VS(30),NS
0088 1 ! -----END SVP1-----
0089 1
0090 1 !  VARBL  SIZE      PURPOSE                                TYPE      RANGE
0091 1 !  -----
0092 1 !  I          COUNTER                                      INTEGER*2
0093 1 !  INSSP      INPUTTED SSP                                INTEGER*2
0094 1 !  IERROR    ERROR FLAG FOR METRIC                      INTEGER*2
0095 1 !  J          COUNTER                                      INTEGER*2
0096 1 !  JANS      OPERATOR RESPONSE FOR LAST BT              INTEGER*2
0097 1 !  L          OPERATOR RESPONSE FOR EDIT BT              INTEGER*2  Y OR N
0098 1 !  NBT       NUMBER OF BT POINTS                        INTEGER*2
0099 1
0100 1 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0101
0102      INTEGER*2 I,IERROR,INSSP,J,JANS,L,NBT
0103
0104 10      IERROR = 0      ! INITIALIZE ERROR FLAG
0105      CALL ICLR          ! CLEAR SCREEN
0106      WRITE(5,2000)      ! USE LAST BT PROMPT
0107      READ(5,1050) JANS  ! OPERATOR RESPONSE
0108      IF(JANS.EQ.'Y') THEN ! IF TRUE, USE LAST BT
0109          DO 250 I=1,NNBT ! USE BT SAVED
0110              D(I)=DD(I)   ! STORE DEPTH
0111              T(I)=TT(I)   ! STORE TEMPERARURE
0112 250      CONTINUE      ! END DO LOOP
0113      NBT=NNBT          ! NUMBER OF BT POINTS
0114      GOTO 75           ! GO TO OUTPUT
0115      END IF            ! END IF BLOCK
0116      CALL ICLR          ! CLEAR SCREEN
0117      WRITE(5,2200)      ! INPUT BT
0118      DO 50 J=1,25      ! DO 25 TIMES

```

```

0119          WRITE(5,1310) J                ! WRITE LOOP COUNTER
0120          READ(5,1300) D(J),T(J)          ! READ DEPTH & TEMP
0121          IF(J.GT.1.AND.D(J).LE.1.)GO TO 60 ! CHECK FOR LAST ENTRY
0122          50  CONTINUE                    ! END DO LOOP
0123          J=26                             ! SET COUNTER TO 26
0124          60  NBT=J-1                     ! # OF BT = COUNTER - 1
0125          CALL DATE(BTDATE)                ! GET DATE
0126          CALL TIME(BTTIME)               ! GET TIME
0127          75  WRITE(5,1380)               ! INPUT DATA TITLE PROMPT
0128          WRITE(5,1400)                   ! PARAMETER TITLES
0129          WRITE(5,1420) (I,D(I),T(I),I=1,NBT) ! DEPTH AND TEMP OR SS
0130          WRITE(5,1500)                   ! CHECK ENTRIES FOR ERRORS
0131          READ(5,1050) L                   ! EDIT DATA RESPONSE
0132          IF(L.EQ.'Y') CALL EDITBT(INSSP,NBT,D,T) ! CORRECT BT DATA
0133          DO 100 I=1,NBT                   ! SAVE BT
0134             DD(I)=D(I)                   ! STORE DEPTH
0135             TT(I)=T(I)                   ! STORE TEMPERATURE
0136          100  CONTINUE                    ! END DO LOOP
0137          NNBT=NBT                         ! NUMBER OF BT POINTS
0138          CALL METRIC(INSSP,D,T,NBT,Z,C,SLNTY,VS(1),IERROR) ! METRIC CALC
0139          IF(IERROR.EQ.1) GO TO 10         ! ERROR IN DATA INPUT
0140          RETURN                           ! RETURN TO CALLING ROUTINE
0141
0142          !-----FORMAT STATEMENTS-----
0143          1050  FORMAT(A1)
0144          1300  FORMAT(2F10.2)
0145          1310  FORMAT(T5,I5,T22,'****',T32,$)
0146          1380  FORMAT(1H /T26,'OPERATOR INPUT DATA')
0147          1400  FORMAT(//T22,'NO.',T32,'DEPTH',T42,'TEMP'
0148                1  /T43,'OR'/T42,'SOUND'/T42,'SPEED'/)
0149          1420  FORMAT(T23,I2,T32,F7.1,T42,F6.1)
0150          1500  FORMAT(1H0/1H$,4X,'DO YOU WISH TO EDIT THE DATA? YES OR NO ',
0151                1  T60,' ')
0152          2000  FORMAT(' DO YOU WANT TO USE THE LAST BT? ',,$)
0153          2200  FORMAT(T20,'ENTER BT IN METRIC AND/OR ENGLISH UNITS'
0154                1  ' (25 POINTS MAX)')
0155          2      /T9,'(TEMPERATURES AND SOUND SPEEDS MAY BE MIXED)')
0156          3      /T9,'(AN EXTRA (CR) TERMINATES ENTRIES)')
0157          4      //T32,'DEPTH',T42,'TEMP'/T43,'OR'/T42,'SOUND'
0158          5      /T42,'SPEED')
0159          END

```

```

0001      SUBROUTINE CLIPOS(IX,IY,I,L1)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: CLIPOS
0005      ! AUTHOR: J. CASCIO, W. WACHTER(FORTRAN 77), NUSC/NL, CODE 3333
0006      ! DATE: 1981 & 9/84 (FORTRAN 77)
0007      ! FUNCTION: THIS SUBROUTINE IS DESIGNED TO SEE IF THE IX OR IY
0008      !              CURSOR POSITION IS IN THE CLIPPED AREA AND IF OUT
0009      !              FIND OUT WHAT QUADRANT THE CURSOR IS IN.
0010      ! INPUTS: CURSOR POSITION
0011      ! OUTPUTS: QUADRANT CURSOR IS IN IF OUT OF CLIP
0012      ! MODULES CALLED: NONE
0013      ! CALLED BY: DRAW, PLUS, POINT, SBOX
0014      ! NOTE: THE NEGATIVE NUMBERS TELL THE CALLING PROGRAM THE VECTOR IS IN A
0015      !       CORNER.
0016
0017      !
0018      !
0019      !
0020      !
0021      !
0022      !
0023      !
0024      !
0025      !
0026      !
0027      !
0028      !
0029      !
0030      !
0031      ! INCLUDE 'SCREEN.INC'
0032      ! -----SCREEN-----
0033      ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0034      ! -----
0035      ! ICLIP  (4)    CLIP BOUNDARIES                        INTEGER*2
0036      ! ISCLIP          CLIPPING FLAG                        LOGICAL*2  TRUE FALSE
0037      ! LENX          LENGTH OF X GRAPHICS BOUNDARY          INTEGER*2
0038      ! LENY          LENGTH OF X GRAPHICS BOUNDARY          INTEGER*2
0039      ! MAXX          MAXIMUM X GRAPHICS BOUNDARY             INTEGER*2
0040      ! MAXY          MAXIMUM Y GRAPHICS BOUNDARY             INTEGER*2
0041      ! MINX          MINIMUM X GRAPHICS BOUNDARY             INTEGER*2
0042      ! MINY          MINIMUM Y GRAPHICS BOUNDARY             INTEGER*2
0043      !
0044      ! INTEGER*2 ICLIP,LENX,LENY
0045      ! INTEGER*2 MAXX,MAXY,MINX,MINY
0046      ! LOGICAL*2 ISCLIP
0047      !
0048      ! COMMON /SCREEN/MINX,MAXX,MINY,MAXY,LENX,LENY,ICLIP(4),ISCLIP
0049      ! -----SCREEN END-----
0050      !
0051      ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0052      ! -----
0053      ! I          LOOP COUNTER                                INTEGER*2
0054      ! TMP          FACTOR IN QUAD1 AND QUAD2 EQUATIONS      INTEGER*2
0055      ! IX          CURSOR X COORDINATE                        INTEGER*2
0056      ! IY          CURSOR Y COORDINATE                        INTEGER*2
0057      ! L          (5) COORDINATES IN OR OUT OF CLIP BOUNDARY LOGICAL*2
0058      ! L1         (4) STORE POINTS OUT OF CLIP AREA          LOGICAL*2
0059      ! QUAD1       QUADRANT                                    INTEGER*2

```



```

0060      ! QUAD2          QUADRANT          INTEGER*2
0061      !
0062      INTEGER*2 I,IX,IY,QUAD1,QUAD2
0063      LOGICAL*2 L(5),L1(4)
0064      REAL*4 TMP
0065
0066      L(1)=(IX.LT.ICLIP(1))      ! X COORDINATE INSIDE CLIP?
0067      L(2)=(IX.GT.ICLIP(2))      ! X COORDINATE INSIDE CLIP?
0068      L(3)=(IY.LT.ICLIP(3))      ! Y COORDINATE INSIDE CLIP?
0069      L(4)=(IY.GT.ICLIP(4))      ! Y COORDINATE INSIDE CLIP?
0070      L(5)=L(1)+L(2)+L(3)+L(4)  ! TRUE IF ALL INSIDE
0071      DO 1 I=1,4                ! DO FOUR TIMES
0072          L1(I)=L(I)            ! STORE POINT OUTSIDE CLIP
0073      1      CONTINUE            ! END DO LOOP
0074      IF (L(5).EQ.-2) THEN        ! CURSOR IN A CORNER
0075          TMP=(ICLIP(1)-ICLIP(2))*1.  ! FACTOR IN EQUATIONS
0076          QUAD1=SIGN(1.,(1.*ICLIP(3)-ICLIP(4))*(IX-ICLIP(1))/TMP+ICLIP(3)
0077          QUAD2=SIGN(1.,(1.*ICLIP(4)-ICLIP(3))*(IX-ICLIP(1))/TMP+ICLIP(4)
0078          L(4)=((QUAD1.EQ.-1) .AND. (QUAD2.LE.0))  ! CHECK THE EXACT QUA
0079          L(3)=((QUAD1.EQ.1) .AND. (QUAD2.GE.0))    ! BY CALCULATING TH
0080          L(2)=((QUAD1.GE.0) .AND. (QUAD2.EQ.-1))  ! OF THE DIAGONAL L
0081          L(1)=((QUAD1.LE.0) .AND. (QUAD2.EQ.1))  ! EXTENDING FROM OP
0082      END IF                      ! CORNERS OF THE CL
0083      DO 3 I=1,4                ! DO FOUR TIMES
0084          IF (L(I)) GOTO 4        ! POINT IS OUT AT L(I)
0085      3      CONTINUE            ! END DO LOOP
0086      I=0                        ! I=0 IF IT IS WITHIN THE PL
0087      4      IF (L(5).EQ.-2) I=-I ! SET I TO NEGATIVE IF IN CO
0088      RETURN                    ! RETURN TO CALLING ROUTINE
0089      END                        ! END SUBROUTINE

```

```

0001      SUBROUTINE CLIPT(IX1,IY1,IX2,IY2,I,J)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: CLIPT
0005      ! AUTHOR: J. CASCIO, W. WACHTER(FORTRAN 77), NUSC/NL, CODE 3333
0006      ! DATE: 1981 & 9/84 (FORTRAN 77)
0007      ! FUNCTION: THIS SUBROUTINE IS DESIGNED TO SET UP PSUEDO-POINTS
0008      !              FOR DRWABS WHICH WILL SHOW UP IN THE CLIP AREA. THE
0009      !              PURPOSE OF CLIPT IS TO PREVENT VECTORS FROM BEING DRAWN
0010      !              OUT OF THE CLIP BOUNDARY WHILE ALLOWING THE BEAM POSITION
0011      !              TO BE UPDATED ACROSS CLIP BOUNDARIES TO THE CURRENT BEAM
0012      !              POSITION.
0013      ! INPUTS: CLIP BOUNDARIES
0014      ! OUTPUTS: UPDATED BEAM POSITION
0015      ! MODULES CALLED: NONE
0016      ! CALLED BY: DRAW
0017      !
0018      INCLUDE 'SCREEN.INC'
0019      1 !-----SCREEN-----
0020      1 ! VARBL   SIZE   PURPOSE                                TYPE      RANGE
0021      1 !-----
0022      1 ! ICLIP   (4)    CLIP BOUNDARIES                        INTEGER*2
0023      1 ! ISCLIP                CLIPPING FLAG                  INTEGER*2  TRUE FALSE
0024      1 ! LENX                LENGTH OF X GRAPHICS BOUNDARY    INTEGER*2
0025      1 ! LENY                LENGTH OF X GRAPHICS BOUNDARY    INTEGER*2
0026      1 ! MAXX                MAXIMUM X GRAPHICS BOUNDARY      INTEGER*2
0027      1 ! MAXY                MAXIMUM Y GRAPHICS BOUNDARY      INTEGER*2
0028      1 ! MINX                MINIMUM X GRAPHICS BOUNDARY      INTEGER*2
0029      1 ! MINY                MINIMUM Y GRAPHICS BOUNDARY      INTEGER*2
0030      1
0031      1      INTEGER*2 ICLIP,LENX,LENY
0032      1      INTEGER*2 MAXX,MAXY,MINX,MINY
0033      1      INTEGER*2 ISCLIP
0034      1
0035      1      COMMON /SCREEN/MINX,MAXX,MINY,MAXY,LENX,LENY,ICLIP(4),ISCLIP
0036      1 !-----SCREEN END-----
0037      !
0038      ! VARBL   SIZE PURPOSE                                TYPE      RANGE
0039      !-----
0040      ! I                PREVIOUS BEAM POSITION                INTEGER*2
0041      ! IDRWX1           STARTING X COORDINATE CLIPPED          INTEGER*2
0042      ! IDRWX2           ENDING X COORDINATE CLIPPED            INTEGER*2
0043      ! IDRWY1           STARTING Y COORDINATE CLIPPED          INTEGER*2
0044      ! IDRWY2           ENDING Y COORDINATE CLIPPED            INTEGER*2
0045      ! IPT              ARRAY POINTER                          INTEGER*2
0046      ! IX1              X COORD OF PREVIOUS BEAM POSIT FUNCTIOININTEGER*2
0047      ! IX2              X COORD OF CURRENT BEAM POSIT FUNCTION INTEGER*2
0048      ! IXPT             X COORDINATE FOR SLOPE CALCULATION     INTEGER*2
0049      ! IYPT             Y COORDINATE FOR SLOPE CALCULATION     INTEGER*2
0050      ! IY1              Y COORDINATE OF PREVIOUS BEAM POSITION  INTEGER*2
0051      ! IY2              Y COORDINATE OF CURRENT BEAM POSITION  INTEGER*2
0052      ! J                CURRENT BEAM POSITION                   INTEGER*2
0053
0054      INTEGER*2 I,IDRWX1,IDRWX2,IDRWY1,IDRWY2,IPT,IX1,IX2
0055      INTEGER*2 IXPT,IYPT,IY1,IY2,J
0056
0057      !-----FUNCTIONS TO DETERMINE THE S
0058      IXPT(IPT)=IX1+ININT((FLOATI(ICLIP(IPT))-FLOATI(IY1))*(FLOATI(IX2
0059      *          FLOATI(IX1))/(FLOATI(IY2)-FLOATI(IY1))))

```

```

0060      IYPT(IPT)=IY1+ININT((FLOATI(ICLIP(IPT))-FLOATI(IX1))*(FLOATI(IY2
0061      *      FLOATI(IY1))/(FLOATI(IX2)-FLOATI(IX1)))
0062 )
0063 !-----PRELIMINARIES-----
0064      IF (I+J.EQ.0) THEN                ! NO CLIPPING NEEDED
0065      TYPE 1                             ! TYPE ONE
0066      GO TO 999                         ! RETURN TO CALLING ROUTINE
0067      END IF                             ! END IF BLOCK
0068      IDRWX1=IX1                        ! STARTING X COORDINATE
0069      IDRWY1=IY1                        ! STARTING Y COORDINATE
0070      IDRWX2=IX2                        ! ENDING X COORDINATE
0071      IDRWY2=IY2                        ! ENDING Y COORDINATE
0072
0073 !-----CLIP THE PREVIOUS END-----
0074      IF (I.NE.0) THEN                  ! NOT WITHIN CLIP BOUNDARIES
0075      IF (I.LE.2) THEN                  ! 2,1,-1,-2,-3,-4
0076      IDRWX1=ICLIP(I)                  ! CLIP STARTING X COORDINATE
0077      IF (IX1.NE.IX2) IDRWY1=IYPT(I)  ! CLIP STARTING Y COORDINATE
0078      ELSE                              ! 3,4
0079      IF (IY1.NE.IY2) IDRWX1=IXPT(I)  ! CLIP STARTING X COORDINATE
0080      IDRWY1=ICLIP(I)                  ! CLIP STARTING Y COORDINATE
0081      END IF                           ! END IF BLOCK
0082      END IF                           ! END IF BLOCK
0083
0084 !-----CLIP THE CURRENT END-----
0085      IF (J.NE.0) THEN                  ! NOT WITHIN CLIP BOUNDARIES
0086      IF (J.LE.2) THEN                  ! 2,1,0,-1,-2,-3,-4
0087      IDRWX2=ICLIP(J)                  ! CLIP ENDING X COORDINATE
0088      IF (IX1.NE.IX2) IDRWY2=IYPT(J)  ! CLIP ENDING Y COORDINATE
0089      ELSE                              ! 3,4
0090      IF (IY1.NE.IY2) IDRWX2=IXPT(J)  ! CLIP ENDING X COORDINATE
0091      IDRWY2=ICLIP(J)                  ! CLIP ENDING Y COORDINATE
0092      END IF                           ! END IF BLOCK
0093      END IF                           ! END IF BLOCK
0094      IX1=IDRWX1                        ! STARTING X COORDINATE
0095      IY1=IDRWY1                        ! STARTING Y COORDINATE
0096      IX2=IDRWX2                        ! ENDING X COORDINATE
0097      IY2=IDRWY2                        ! ENDING Y COORDINATE
0098      999 RETURN                        ! RETURN TO CALLING ROUTINE
0099
0100 !-----FORMAT STATEMENT-----
0101 1      FORMAT(' ** ERROR IN 'CLIPT', AT LEAST ONE END MUST BE ',
0102      +      'CLIPPED **')
0103      END

```

```

0001          SUBROUTINE CNNCT(SEG,PNDX,R)
0002      !
0003      ! PROLOGUE:
0004      ! MODULE NAME: CNNCT
0005      ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006      ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007      ! DATE: 1982 & 6/84 (FORTRAN 77)
0008      ! FUNCTION: THIS SUBROUTINE IS DESIGNED TO SELECT SORT AND CONNECT UP
0009      !               POLYGONS FROM THE FOUR QUADRANTS OF DATA READ FROM THE DATA
0010      !               BASE. ADJACENT POLYGONS WITH THE SAME CODE ARE THEN CONNECTED
0011      !               TO FORM ONE WHICH WILL BE DISPLAYED. THE POLYGON ARE SORTED
0012      !               ACCORDING TO THEIR DISTANCE FROM THE SHIP.
0013      ! INPUTS:
0014      ! OUTPUTS:
0015      ! MODULES CALLED: END1, END2, KMOD
0016      ! CALLED BY: CRUNCH
0017      !
0018          INCLUDE 'MAP.PAR'
0019      1      PARAMETER STOLEN=3800
0020      1      PARAMETER SEGLEN=60, POLLEN=40
0021      1      PARAMETER WRKLEN=1000, NDXLEN=300
0022      1      PARAMETER MAXDTY=3
0023      1      PARAMETER TOL=3
0024      1      PARAMETER DEG=57.2957795
0025      1      PARAMETER RAD=.017453293
0026      1      PARAMETER PI=3.14159265
0027      1      PARAMETER ERAD=3440.3
0028      1      PARAMETER S251=63001
0029      1      PARAMETER TWO15=32768
0030      1
0031      1 !      INTEGER*2 MAXDTY,NDXLEN,POLLEN,SEGLEN,STOLEN,TOL,WRKLEN
0032      1 !      INTEGER*4 S251,TWO15
0033      1 !      REAL*4      DEG,ERAD,PI,RAD
0034      1      INCLUDE 'CS.INC'
0035      1 ! -----CS-----
0036      1 ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0037      1 ! -----
0038      1 ! S      -1,3800    POLYGON AND SEGMENT STORAGE ARRAY      REAL*4
0039      1 ! STOLEN          STORAGE ARRAY LENGTH (FOR SEGS & POLYS)  PARM
0040      1 !
0041      1      REAL*4  S(-1:STOLEN)
0042      1
0043      1      COMMON /CS/      S
0044      1 ! -----CS-END-----
0045      1
0046      !
0047      ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0048      ! -----
0049      ! END1          USER FUNCTION                                INTEGER*2
0050      ! END2          USER FUNCTION                                INTEGER*2
0051      ! F            FIRST ENDPOINT FOUND FLAG                    BYTE
0052      ! I            LOOP COUNTER                                INTEGER*2
0053      ! II           LOOP COUNTER                                INTEGER*2
0054      ! ISAVE        SAVE INDEX OF NEW POLYGON                    INTEGER*2
0055      ! J            LOOP COUNTER                                INTEGER*2
0056      ! JJ           LOOP COUNTER                                INTEGER*2
0057      ! K            LOOP COUNTER                                INTEGER*2
0058      ! KK           LOOP COUNTER                                INTEGER*2
0059      ! KMOD         USER FUNCTION                                INTEGER*2

```

```

0060      ! N                POLYGON N                INTEGER*2
0061      ! NDX              INDEX                    INTEGER*2
0062      ! NN               POLYGON NN               INTEGER*2
0063      ! PNDX (0,160)    POLYGON INDEX ARRAY       INTEGER*2
0064      ! PP1              SEGMENT SECOND ENDPOINT   INTEGER*2
0065      ! PP2              SEGMENT SECOND ENDPOINT   INTEGER*2
0066      ! P1               SEGMENT FIRST ENDPOINT    INTEGER*2
0067      ! P2               SEGMENT FIRST ENDPOINT    INTEGER*2
0068      ! R                ERROR RETURN FLAG         BYTE
0069      ! SEG              1 OF 4 INTERIOR BORDER SEGS  INTEGER*2      2,5,8,11
0070      !
0071      ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0072
0073      INTEGER*2 I,II,ISAVE,J,JJ,K,KK,N,NDX,NN
0074      INTEGER*2 PNDX(0:4*POLLEN),PP1,PP2,P1,P2,SEG
0075      INTEGER*2 END1,END2,KMOD      !FUNCTIONS
0076      BYTE      R,F
0077
0078      R=.FALSE.                ! SET ERROR FLAG
0079      IF (PNDX(0).GT.4*POLLEN) GOTO 101 ! CHECK TOTAL NUM OF POLS
0080
0081      !-----PROCESS EACH POL EXCEPT LAST
0082      DO 14 I=1,PNDX(0)-1      ! PROCESS ALL BUT LAST POL
0083          N=PNDX(I)            ! SET POLYGON N
0084          DO 13 J=1,S(N+1)      ! DO EACH SEG IN POL N
0085              IF (ABS(S(N+J+2)).NE.SEG .OR. S(N+1).LT.1) ! NOT INTERIOR
0086                  *          GOTO 13 ! BORDER OR < 1
0087                  F=.FALSE.      ! 1ST ENDPT FOUND FLAG
0088                  DO 12 II=I+1,PNDX(0) ! ALL POLS AFTER N
0089                      NN=PNDX(II) ! SET POLYGON NN
0090                      DO 11 JJ=1,S(NN+1) ! DO EACH SEG IN POL NN
0091                          IF (S(NN+1).LT.1. .OR. S(NN+JJ+2).NE. -S(N+J+2))
0092                              *          GOTO 11 ! NO SEGS IN POLYGON
0093                              IF (.NOT.F) THEN ! NOT 1ST ENDPOINT
0094                                  P1=END1(N,J) ! ASSIGN FIRST ENDPOINT
0095                                  P2=END2(N,J) ! ASSIGN FIRST ENDPOINT
0096                                  F=.TRUE.      ! 1ST ENDPT FOUND FLAG
0097                                  END IF      ! END IF BLOCK
0098                                  PP1=END1(NN,JJ) ! ASSIGN SECOND ENDPOINT
0099                                  PP2=END2(NN,JJ) ! ASSIGN SECOND ENDPOINT
0100                                  IF (ABS(P1-PP2).GT.TOL .OR. ABS(P2-PP1).GT.TOL
0101                                      *          .OR. S(N).NE.S(NN)) GOTO 11 ! DO NEXT SEGMENT
0102
0103      !-----CONNECT TWO POLYGONS-----
0104      NDX=S(0)                ! SET LENGTH OF ARRAY
0105      IF (S(N+1)+S(NN+1)+S(N+2)+S(NN+2)+1.GT.
0106          *          S(-1)-NDX) GOTO 103 ! CHECK FOR OVERFLOW
0107      ISAVE=NDX+1             ! SAVE INDEX OF NEW POL
0108      S(NDX+1)=S(N)           ! STORE MGS/SSP CODE
0109      S(NDX+2)=S(N+1)+S(NN+1)-2 ! STORE COMBINED SEG COUNT
0110      S(NDX+3)=S(N+2)+S(NN+2) ! STORE COMB LABEL COUNT
0111      NDX=NDX+3               ! INCREMENT INDEX
0112      DO 4 K=1,S(N+1)         ! STORE SEGS FROM POL N
0113          IF (K.NE.J) THEN    ! NOT = SEG IN OUTER LOOP
0114              NDX=NDX+1        ! INCREMENT INDEX
0115              S(NDX)=S(N+K+2) ! STORE SEG FROM POL N
0116          ELSE                ! K = J
0117              DO 3 KK=JJ+1,JJ+S(NN+1)-1 ! STORE SEGS FROM POL NN
0118                  NDX=NDX+1 ! INCREMENT INDEX

```

```

0119          S(NDX)=S(2+NN+(KMOD(KK-1,IIFIX(S(NN+1))))+1))
0120      3          CONTINUE          ! END DO LOOP
0121      )121      END IF          ! END IF BLOCK
0122      4          CONTINUE          ! END DO LOOP
0123      DO 5 K=1,S(ISAVE+1)          ! CHECK ADJACENT BORDERS
0124      KK=KMOD(K,IIFIX(S(ISAVE+1)))+1 ! POINTER
0125      IF (S(ISAVE+K+2).EQ.-S(ISAVE+KK+2)) GOTO 6!EXIT LOOP
0126      5          CONTINUE          ! END DO LOOP
0127      GOTO 8          ! SKIP NEXT
0128
0129      !-----ELIMINATE ADJACENT INERIOR BORDER SEGS
0130      6          S(ISAVE+K+2)=0.          ! ZERO OUT STORAGE
0131      S(ISAVE+KK+2)=0.          ! ZERO OUT STORAGE
0132      KK=0          ! RESET POINTER
0133      DO 7 K=ISAVE+3,ISAVE+S(ISAVE+1)+2 ! ELIMINATE SEGMENT
0134      IF (S(K).NE.0.) S(K-KK)=S(K) ! RESET STORAGE ARRAY
0135      IF (S(K).EQ.0.) KK=KK+1 ! RESET POINTER
0136      7          CONTINUE          ! CONTINUE
0137      S(ISAVE+1)=S(ISAVE+1)-2          ! RESET STORAGE ARRAY
0138      NDX=NDX-2          ! RESET INDEX
0139
0140      !-----STORE LABELS FROM N----
0141      8          DO 9 K=N+3+S(N+1),N+2+S(N+1)+S(N+2) ! FOR LABELS
0142      IF (S(N+2).GE.1.) THEN          ! STORED >= 1
0143      NDX=NDX+1          ! INCREMENT INDEX
0144      S(NDX)=S(K)          ! STORE LABELS FROM N
0145      END IF          ! END IF BLOCK
0146      9          CONTINUE          ! END DO LOOP
0147
0148      !-----STORE LABELS FROM NN-----
0149      DO 10 KK=NN+3+S(NN+1),NN+2+S(NN+1)+S(NN+2)! FOR LABELS
0150      IF (S(NN+2).GE.1.) THEN          ! STORED >= 1
0151      NDX=NDX+1          ! INCREMENT INDEX
0152      S(NDX)=S(KK)          ! STORE LABELS FROM NN
0153      END IF          ! END IF BLOCK
0154      10          CONTINUE          ! END DO LOOP
0155      PNDX(I)=0          ! DELETE POL REFERENCE N
0156      PNDX(II)=ISAVE          ! STORE NEW POL ON TOP OF NN
0157      S(0)=NDX          ! UPDATE USED LENGTH OF S
0158      GOTO 14          ! EXIT LOOPS
0159      11          CONTINUE          ! END DO LOOP
0160      12          CONTINUE          ! END DO LOOP
0161      13          CONTINUE          ! END DO LOOP
0162      14          CONTINUE          ! END DO LOOP
0163      GO TO 999          ! TO CALLING ROUTINE
0164
0165      !-----ERROR STATEMENTS-----
0166      101      WRITE(5,102)          ! POL INDEX ARRAY OVERFLOW
0167      R=.TRUE.          ! SET ERROR FLAG
0168      GO TO 999          ! RETURN TO CALLING ROUTINE
0169      103      WRITE(5,104)          ! STORAGE ARRAY OVERFLOW
0170      R=.TRUE.          ! SET ERROR FLAG
0171      999      RETURN          ! TO CALLING ROUTINE
0172
0173      !-----FORMAT STATEMENTS-----
0174      102      FORMAT(X,'POLYGON INDEX ARRAY OVERFLOW IN CNNECT')
0175      104      FORMAT(X,'STORAGE ARRAY OVERFLOW IN CNNECT')
0176      END

```

```

0001      SUBROUTINE CONECT(MX,MY,NEWX,NEWY)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: CONECT
0005      ! AUTHOR: J. CASCIO, W. WACHTER(FORTRAN 77), NUSC/NL, CODE 3333
0006      ! DATE: 1981 & 9/84 (FORTRAN 77)
0007      ! FUNCTION: DRAWS A VECTOR BETWEEN THE POINTS (MX,MY) AND (NEWX,NEWY)
0008      !           SETS CURRENT POSITION TO (NEWX,NEWY)
0009      ! INPUTS: COORDINATES OF TWO POINTS TO BE CONNECTED
0010      ! OUTPUTS: VECTOR DRAWN BETWEEN POINTS
0011      ! MODULES CALLED: NONE
0012      ! CALLED BY: CONCTU, SVPGRF
0013      !
0014      INCLUDE 'TK4025.INC'
0015 1 ! -----TK4025-----
0016 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0017 1 ! -----
0018 1 ! KBEAMX          CURRENT BEAM X POSITION              INTEGER*2
0019 1 ! KBEAMY          CURRENT BEAM Y POSITION              INTEGER*2
0020 1
0021 1      INTEGER*2 KBEAMX,KBEAMY
0022 1
0023 1      COMMON/TK4025/KBEAMX,KBEAMY
0024 1 ! -----TK4025 END-----
0025 1
0026 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0027 1 ! -----
0028 1 ! MX          STARTING X COORDINATE                    INTEGER*2
0029 1 ! MY          STARTING Y COORDINATE                    INTEGER*2
0030 1 ! NEWX        ENDING X COORDINATE                     INTEGER*2
0031 1 ! NEWY        ENDING Y COORDINATE                     INTEGER*2
0032 1
0033 1      INTEGER*2 MX,MY,NEWX,NEWY
0034 1
0035 1      WRITE(5,1) MX,MY,NEWX,NEWY      ! PUT COMMAND OUT TO THE TERMINAL
0036 1      KBEAMX=NEWX                    ! UPDATE X CURRENT POSITION
0037 1      KBEAMY=NEWY                    ! UPDATE Y CURRENT POSITION
0038 1      RETURN                          ! RETURN TO CALLING ROUTINE
0039 1
0040 1 !-----FORMAT STATEMENTS-----
0041 1      1 FORMAT(' !VEC ',4I5)
0042 1      END

```

```

0001      SUBROUTINE CRUNCH(RECNDX,INFO,DTYPE,IY,IX,R)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: CRUNCH
0005      ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006      ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007      ! DATE: 1982 & 6/84 (FORTRAN 77)
0008      ! FUNCTION:
0009      !     THIS SUBROUTINE IS DESIGNED TO:
0010      !         1) CHECK TO SEE IF DATA FOR EACH QUADRANT EXISTS
0011      !         2) CALL OTHER SUBROUTINES TO GET DATA, CONNECT SEGMENTS,
0012      !         AND TAKE CARE OF DOUGHNUT MIDDLES OF CONNECTED POLYGONS
0013      !         3) COMPILE POLYGON OR SEGMENT LISTS
0014      !         4) CALL SUBROUTINE TO DO ROTATIONAL ANALYSIS ON POLY-
0015      !         GONS OR DISTANCE ANALYSIS ON SEGMENTS
0016      !         5) SORT THE POLYGONS OR SEGMENTS ACCORDING TO DISTANCE
0017      !         6) RETURN CODES AND DISTANCES
0018      ! INPUTS: HARD COPY SELECTION, OPERATOR SELECTION TO UPDATE
0019      !     PARAMETERS OR NOT. VARIABLES IN COMMONS.
0020      ! OUTPUTS: CRT PROMPTING MESSAGES TO OPERATOR
0021      ! MODULES CALLED: BMOD, CNNCT, DNUT, GETREC, INDX, OPNFIL, PDIST
0022      ! CALLED BY: MAP
0023      !
0024      PARAMETER MINNUM=1.E-20
0025      PARAMETER MAXNUM=1.E21
0026      PARAMETER EPSLN=1.E-5
0027      INCLUDE 'MAP.PAR'
0028      1  PARAMETER STOLEN=3800
0029      1  PARAMETER SEGLEN=60, POLLEN=40
0030      1  PARAMETER WRKLEN=1000, NDXLEN=300
0031      1  PARAMETER MAXDTY=3
0032      1  PARAMETER TOL=3
0033      1  PARAMETER DEG=57.2957795
0034      1  PARAMETER RAD=.017453293
0035      1  PARAMETER PI=3.14159265
0036      1  PARAMETER ERAD=3440.3
0037      1  PARAMETER S251=63001
0038      1  PARAMETER TWO15=32768
0039      INCLUDE 'CFILE.INC'
0040      1  ! -----CFILE.INC-----
0041      1  ! VARBL  SIZE  PURPOSE  TYPE  RANGE
0042      1  ! -----
0043      1  ! FNAME  (21)  MAP FILE NAME  CHAR
0044      1  ! OPEN    OPEN FLAG  LOGICAL*1 .FALSE.
0045      1  !
0046      1  LOGICAL*1 OPEN
0047      1  CHARACTER*1 FNAME(21)
0048      1
0049      1  COMMON /CFILE/ OPEN,FNAME
0050      1  ! -----END CFILE.INC-----
0051      1
0052      INCLUDE 'CL.INC'
0053      1  ! -----CL.INC-----
0054      1  ! VARBL  SIZE  PURPOSE  TYPE  RANGE
0055      1  ! -----
0056      1  ! LATMAX  MAXIMUM LATIITUDE  INTEGER*2
0057      1  ! LATMIN  MINIMUM LATIITUDE  INTEGER*2
0058      1  ! LNGMAX  MAXIMUM LONGITUDE  INTEGER*2
0059      1  ! LNGMIN  MINIMUM LONGITUDE  INTEGER*2

```



```

0060 1 !
0061 1      INTEGER*2  LATMIN,LATMAX,LNGMIN,LNGMAX
0062 1
0063 1      COMMON /CL/   LATMIN,LATMAX,LNGMIN,LNGMAX
0064 1 ! -----END CL.INC-----
0065 1
0066      INCLUDE 'CLOC.INC'
0067 1 ! -----CLOC.INC-----
0068 1 !  VARBL  SIZE      PURPOSE                                TYPE  RANGE
0069 1 !  -----
0070 1 !  BLAT      BASE LATITUDE                                REAL*4
0071 1 !  BLNG      BASE LONGITUDE                                REAL*4
0072 1 !  LAT       LATITUDE OF SHIP'S LOCATION                REAL*4
0073 1 !  LNG       LONGITUDE OF SHIP'S LOCATION                REAL*4
0074 1 !  NMLT50    # OF NAUTICAL MILES PER 50TH DEGREE        REAL*4
0075 1 !              OF LATITUDE
0076 1 !  NMLG50    # OF NAUTICAL MILES PER 50TH DEGREE        REAL*4
0077 1 !              OF LONGITUDE
0078 1 !
0079 1      REAL*4  LAT,LNG,BLAT,BLNG,NMLT50,NMLG50
0080 1
0081 1      COMMON /CLOC/   LAT,LNG,BLAT,BLNG,NMLT50,NMLG50
0082 1 ! -----END CLOC.INC-----
0083      INCLUDE 'CLOG.INC'
0084 1 ! -----CLOG.INC-----
0085 1 !  VARBL  SIZE      PURPOSE                                TYPE  RANGE
0086 1 !  -----
0087 1 !  CNVRT(-1:0)
0088 1 !  DG
0089 1 !  DL
0090 1 !
0091 1      BYTE    CNVRT(-1:0),DG,DL
0092 1
0093 1      COMMON /CLOG/   CNVRT,DL,DG
0094 1 ! -----END CLOG.INC-----
0095      INCLUDE 'CS.INC'
0096 1 ! -----CS-----
0097 1 !  VARBL  SIZE      PURPOSE                                TYPE  RANGE
0098 1 !  -----
0099 1 !  S      -1,3800    POLYGON AND SEGMENT STORAGE ARRAY        REAL*4
0100 1 !  STOLEN          STORAGE ARRAY LENGTH (FOR SEGS & POLYS)  PARM
0101 1 !
0102 1      REAL*4  S(-1:STOLEN)
0103 1
0104 1      COMMON /CS/     S
0105 1 ! -----CS-END-----
0106 1
0107 1
0108 1 !  VARBL  SIZE      PURPOSE                                TYPE  RANGE
0109 1 !  -----
0110 1 !  AFILE      FILE TO BE OPENED                                BYTE
0111 1 !  AX         X COORDINATE OF POINT A                          REAL*4
0112 1 !  AY         Y COORDINATE OF POINT A                          REAL*4
0113 1 !  B          POINT B                                          REAL*4
0114 1 !  BX         X COORDINATE OF POINT B                          REAL*4
0115 1 !  BY         Y COORDINATE OF POINT B                          REAL*4
0116 1 !  DA         BOTTOM DEPTH AT CLOSEST POINT                    REAL*4
0117 1 !  DISTAB     DISTANCE FROM A TO B                              REAL*4
0118 1 !  DISTAT     DISTANCE FROM A TO T                              REAL*4

```

```

0119      ! DISTBT          DISTANCE FROM B TO T          REAL*4
0120      ! DL             ROTATION FOR POLYGON FLAG      BYTE
0121      ! DTYPE          DATA TYPE BEING USED          INTEGER*2
0122      ! I              LOOP INDEX                     INTEGER*2
0123      ! IB            (4)  CONSTANTS  2,5,8,11         REAL*4      2,5,8,11
0124      ! INFO           INFO PASSED TO THE MAIN PROGRAM REAL*4
0125      ! INDX           FUNCTION                        INTEGER*2
0126      ! IX            (4)  POSITION OF THE CURRENT QUADRANT INTEGER*2
0127      ! IY            (4)  POSITION OF THE CURRENT QUADRANT INTEGER*2
0128      ! J              LOOP INDEX                     INTEGER*2
0129      ! K              TEMPORARY STORAGE               REAL*4
0130      ! NDX            DATA BASE INDEX                INTEGER*4
0131      ! PDIST          FUNCTION                        INTEGER*2
0132      ! PNDX          (0:160) POLYGON INDEX            INTEGER*2
0133      ! R              ERROR FLAG                      BYTE
0134      ! RECNDX         ()  POINTERS INTO THE DATA BASE INTEGER*2
0135      ! SLOPAB          SLOPE OF SEGMENT AB             REAL*4
0136      ! SLOPTS         SLOPE                          REAL*4
0137      ! SNDX           (0:240) SEGEMNT INDEX            INTEGER*2
0138      ! T              POINT T                         REAL*4
0139      ! TX             X COORDINATE OF POINT T          REAL*4
0140      ! TY             Y COORDINATE OF POINT T          REAL*4
0141      ! T1             LENGTH OF POLYGON/SEGMENT LIST   INTEGER*2
0142      ! T2             LENGTH OF POLYGON/SEGMENT LIST   INTEGER*2
0143      !
0144      ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0145
0146      REAL*4  AX,AY,BX,BY, TX, TY, DA, DB, DISTAB, DISTAT, DISTBT, B
0147      REAL*4  SLOPAB, SLOPTS, INFO, K, T, BMOD, BDIV
0148      INTEGER*4 NDX
0149      INTEGER*2 I, IB(4), INDX, IX(4), IY(4), SNDX(0:4*SEGLN)
0150      INTEGER*2 PDIST, PNDX(0:4*POLLEN), T1, T2, J
0151      INTEGER*2 RECNDX((LATMAX-LATMIN)*(LNGMAX-LNGMIN)*MAXDTY)
0152      INTEGER*2 DTYPE
0153      BYTE     AFILE, R
0154      DATA    IB      /2,5,8,11/
0155      !
0156      !-----INITIALIZATION-----
0157      BDIV(B,I)=AINT(B/FLOAT(I))      ! TRUNCATING DIVISION
0158      AFILE=.FALSE.                  ! SET TO DATA ("B") FILE
0159      R=.FALSE.                      ! SET ERROR FLAG
0160      S(0)=12.                       ! NUMBER INITIAL EDGE SEGMENTS
0161      DO 1 I=0,4*SEGLN               ! FOR ALL SEGMENTS
0162          SNDX(I)=0                  ! INITIALIZE SEGMENT INDEX
0163      1  CONTINUE                    ! END LOOP
0164      DO 2 I=0,4*POLLEN              ! FOR ALL POLYGONS
0165          PNDX(I)=0                  ! INITIALIZE POLYGON INDEX
0166      2  CONTINUE                    ! END LOOP
0167      IF (.NOT.OPEN) CALL OPNFIL(AFILE,R) ! OPEN FILE
0168      IF (R) GO TO 999                ! IF ERROR IN OPEN, RETURN
0169      !
0170      !-----VALIDATE & PROCESS EACH QUADRANT
0171      DO 3 I=1,4                     ! FOR 5 DEG SQUARES DISPLAYED
0172          NDX=RECNDX(INDX(IY(I),IX(I),DTYPE)) ! GET INDEX POINTER
0173          IF (NDX.EQ.-TWO15) GOTO 3      ! NO DATA FOR THE SQUARE
0174          NDX=NDX+TWO15                 ! OFFSET DATA BASE INDEX
0175          CALL GETREC(SNDX,PNDX,I,DTYPE,NDX,R) ! GET DATA FOR SQUARE
0176          IF (R) GO TO 999                ! IF ERROR IN GETREC, RETURN
0177      3  CONTINUE                      ! END LOOP

```

```

0178             IF (DG) GOTO 6                     ! NOT POLYGON DATA
0179             !
0180             !-----CONNECT ADJACENT POLYGONS-----
0181             DO 4 I=1,4                           ! LOOP TO CONNECT POLY SEGS
0182             CALL CNNCT(IB(I),PNDX,R)             ! CONNECT POLYGONS TOGETHER
0183             IF (R) GO TO 999                       ! IF ERROR IN CONNECT, RETURN
0184             4      CONTINUE                         ! END LOOP
0185             !
0186             !-----ELIMINATE NEWLY CREATED MIDDLES
0187             DO 5 I=1,4                           ! LOOP TO DELETE MIDDLES
0188             CALL DNUOT(IB(I),PNDX,R)             ! ELIMINATE DOUGHNUT MIDDLES
0189             IF (R) GO TO 999                       ! IF ERROR IN DNUOT, RETURN
0190             5      CONTINUE                         ! END LOOP
0191             !
0192             !-----SAVE SEGMENT OR POLYGON LIST
0193             6      T1=IIFIX(S(0))+1                 ! SAVE STORAGE ARRAY INDEX
0194             IF (DG) GOTO 8                         ! IF NOT POLYGON DATA BRANCH
0195             DO 7 I=0,4*POLLEN                     ! COPY POL LIST INTO SEG LIST
0196             SNDX(I)=PNDX(I)                       ! FILL SEGMENT INDEX ARRAY
0197             7      CONTINUE                         ! END LOOP
0198             8      T2=SNDX(0)                       ! SAVE LENGTH OF SEG/POL LIST
0199             J=0                                    ! RESET ZAPPED POLYGON COUNT
0200             DO 9 I=1,T2                           ! THROW OUT ZAPPED POLYGONS
0201             IF (SNDX(I).NE.0) SNDX(I-J)=SNDX(I) ! REMOVE FROM LIST
0202             IF (SNDX(I).EQ.0) J=J+1               ! CORRECT INDEX
0203             9      CONTINUE                         ! END LOOP
0204             T2=T2-J                                ! SAVE LENGTH OF NEW POLLIST
0205             SNDX(0)=T2                             ! STORE # OF SEGMENTS OR POLS
0206             !
0207             !-----STORE POLYGON LIST AND LENGTH-----
0208             IF (2.*T2.GT.(S(-1))-T1) GOTO 101 ! ERROR:STORAGE ARRAY OVERFLOW
0209             S(T1)=FLOATI(T2)                       ! STORE LENGTH OF SEG/POL LIST
0210             DO 10 I=1,T2                           ! STORE SEG/POL LIST,DISTANCES
0211             S(T1+I)=(SNDX(I))                     ! STORE SEGMENT OR POLYGON
0212             IF (DL) S(T1+T2+I)=PDIST(SNDX(I)) ! ROTATION FOR POLS
0213             IF (DG) S(T1+T2+I)=SQRT(S(SNDX(I)+2)) ! DISTANCE FOR SEGS
0214             10     CONTINUE                         ! END LOOP
0215             !
0216             !-----SORT POLYGONS OR SEGMENTS ACCORDING TO DISTANCE FROM SHIP
0217             IF (T2.LE.1) GOTO 15                   ! ONLY ONE SEGEMNT OR POLYGON
0218             11     DO 13 I=T1+T2+1,T1+2*T2-1       ! FOR DISTANCES OF POL OR SEG
0219             DO 12 J=I+1,T1+T2*2                   ! FOR DISTANCES OF POL OR SEG
0220             IF (S(J).LT.S(I)) THEN                 ! DISJOINT POLYGONS
0221             K=S(J-T2)                             ! SWITCH REFERENCE CODES
0222             S(J-T2)=S(I-T2)                       ! SWITCH REFERENCE CODES
0223             S(I-T2)=K                             ! SWITCH REFERENCE CODES
0224             K=S(I)                                ! SWITCH DISTANCES
0225             S(I)=S(J)                             ! SWITCH DISTANCES
0226             S(J)=K                                ! SWITCH DISTANCES
0227             END IF                                ! END IF BLOCK
0228             12     CONTINUE                         ! END DO LOOP
0229             13     CONTINUE                         ! END DO LOOP
0230             !
0231             !-----ELIMINATE MULTIPLE INTERIORITIES OF POLYGON DISTANCES
0232             IF (S(T1+T2+1).GE.0. .OR. S(T1+T2+2).GE.0.) GOTO 15 ! DIST>0
0233             DO 14 I=T1+T2+2,T1+2*T2               ! MULTIPLE INTERIORS CHECK
0234             IF (S(I).GE.0.) GOTO 11                 ! SHIP NOT INTERIOR TO POL
0235             S(I-1)=-S(I)                           ! SET ALL BUT CLOSEST POL DIST
0236             14     CONTINUE                         ! END DO LOOP

```

```

0237          GOTO 11                      ! RESORT POLYGON DISTANCES
0238
0239  !-----SEGMENT DATA-----
0240  15      S(0)=FLOATI(T1)                ! RESET # OF POLS OR SEGS
0241          IF (.NOT.DG) THEN              ! IF POLYGON DATA
0242              INFO=S((S(T1+1)))          ! SET TO MGS OR SSV OF POLYGON
0243              GO TO 999                  ! RETURN TO CALLING ROUTINE
0244          END IF                          ! END ID BLOCK
0245  16      IF ((T2.LT.2) .OR. (ABS(S(S(T1+1)+1)).GT.355) ! ONLY 1 CONTOUR
0246          *      .OR. (S(T1+T2+1).LE.2)) INFO=S(S(T1+1)+3) ! =BOTTOM DEPTH
0247          IF (INFO.GE.0) GO TO 999        ! RETURN TO CALLING ROUTINE
0248          AX=BMOD(S(S(T1+1)+4),501)      ! X DIST OF CLOSEST POINT
0249          AY=BDIV(S(S(T1+1)+4),501)      ! Y DIST OF CLOSEST POINT
0250          DA=S(S(T1+1)+3)                ! BOTTOM DEPTH AT CLOSEST PT
0251          DO 17 I=2,T2                    ! FOR ALL DISTANCES
0252              IF (S(S(T1+I)+3).NE.DA) THEN ! NOT = BOTTOM DEPTH
0253                  BX=BMOD(S(S(T1+I)+4),501) ! X DISTANCES
0254                  BY=BDIV(S(S(T1+I)+4),501) ! Y DISTANCES
0255                  DB=S(S(T1+I)+3)          ! BOTTOM DEPTH
0256                  GOTO 18                  ! EXIT LOOP
0257              END IF                      ! END IF BLOCK
0258  17      CONTINUE                        ! END DO LOOP
0259          BX=BMOD(S(S(T1+2)+4),501)      ! X DISTANCE
0260          BY=BDIV(S(S(T1+2)+4),501)      ! Y DISTANCE
0261          DB=S(S(T1+2)+3)                ! BOTTOM DEPTH
0262
0263  18      DISTAB=SQRT((BX-AX)**2+(BY-AY)**2) ! DISTANCE FROM A TO B
0264          IF ((AX.NE.BX) .AND. (AY.NE.BY)) SLOPAB=(BY-AY)/(BX-AX) !SLOPE
0265          IF (AY.EQ.BY) SLOPAB=MINNUM     ! SLOPE OF SEGMENT AB
0266          IF (AX.EQ.BX) SLOPAB=MAXNUM     ! SLOPE OF SEGMENT AB
0267          SLOPTS=SLOPAB*(-AX)+AY          ! SLOPE
0268          B=LNG/SLOPAB+LAT                ! LOCATION OF B
0269          TX=(B-SLOPTS)/(SLOPAB+1/SLOPAB) ! X DISTANCE
0270          TY=(B*SLOPAB**2+SLOPTS)/(1+SLOPAB**2) ! Y DISTANCE
0271          DISTAT=SQRT((TX-AX)**2+(TY-AY)**2) ! DISTANCE FROM A TO T
0272          DISTBT=SQRT((TX-BX)**2+(TY-BY)**2) ! DISTANCE FROM B TO T
0273          IF (DISTAT+DISTBT.GT.DISTAB+EPSLN) THEN ! COMPARE DISTANCES
0274              IF (DISTAT.LE.DISTBT) THEN ! A TO T <= B TO T
0275                  DISTAT=-AMIN1(DISTAB/2.,DISTAT) ! DISTANCE A TO T
0276              ELSE                        ! A TO T > B TO T
0277                  DISTAT=AMIN1(DISTAT,DISTAB*1.5) ! DISTANCE FROM A TO T
0278              END IF                      ! END IF BLOCK
0279          END IF                          ! END IF BLOCK
0280  20      IF (DISTAB.EQ.0) DISTAB=MINNUM ! MINIMUM DISTANCE A TO B
0281          INFO=10*(ANINT(((DB-DA)*(DISTAT/DISTAB)+DA)/10)) ! INFO
0282          GO TO 999                      ! RETURN TO CALLING ROUTINE
0283
0284  !-----ERRORS-----
0285  101      WRITE(5,102)                  ! ARRAY OVERFLOW ERROR
0286          R=.TRUE.                      ! SET ERROR FLAG TO TRUE
0287  999      RETURN                        ! RETURN TO CALLING ROUTINE
0288
0289  !-----FORMAT STATEMENT-----
0290  102      FORMAT (X,'STORAGE ARRAY OVERFLOW IN CRUNCH')
0291          END

```

```

0001          SUBROUTINE DNUOT(SEG,PNDX,R)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: DNUOT
0005      ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006      ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007      ! DATE: 1982 & 6/84 (FORTRAN 77)
0008      ! FUNCTION: SUBROUTINE DOUGHNUT IS DESIGNED TO:
0009      !             1) DETECT POLYGON DOUGHNUT STRUCTURES
0010      !             2) REMOVE INTERIOR BORDER SECTIONS OF THE POLYGONS
0011      !                 THAT ALLOW THEIR FORMATION
0012      !             3) ELIMINATE DOUGHNUT MIDDLE SEGMENT FROM THE POLYGON
0013      ! INPUTS: POLYGONS WITH DOUGHNUT STRUCTURES
0014      ! OUTPUTS: POLYGONS WITHOUT DOUGHNUT STRUCTURES
0015      ! MODULES CALLED: END1, END2, FNP, KMOD
0016      ! CALLED BY: CRUNCH
0017      !
0018      ! =====
0019      ! NOTE: ALL LOOPS IN THIS SUBROUTINE ARE PRETEST LOOPS THAT ARE
0020      !       MANUALLY CREATED. THIS ALLOWS LOOP EXECUTION TO BE
0021      !       EASILY BY-PASSED.
0022      !
0023      !       J-JJ: DENOTES SUB-POLYGON DEFINED BY SEGMENTS J THRU JJ
0024      !       JJ-J: DENOTES SUB-POLYGON DEFINED BY SEGMENTS JJ THRU J
0025      !           (THE LATTER WRAPS AROUND THE SEGMENT LIST)
0026      !       CROSS POINT: A SEGMENT ENDPOINT ON THE LINE DEFINED BY SEG
0027      ! =====
0028      !
0029      INCLUDE 'MAP.PAR'
0030      )030 1      PARAMETER STOLEN=3800
0031      1      PARAMETER SEGLEN=60, POLLEN=40
0032      1      PARAMETER WRKLEN=1000, NDXLEN=300
0033      1      PARAMETER MAXDTY=3
0034      1      PARAMETER TOL=3
0035      1      PARAMETER DEG=57.2957795
0036      1      PARAMETER RAD=.017453293
0037      1      PARAMETER PI=3.14159265
0038      1      PARAMETER ERAD=3440.3
0039      1      PARAMETER S251=63001
0040      1      PARAMETER TWO15=32768
0041      1
0042      1 !      INTEGER*2 MAXDTY,NDXLEN,POLLEN,SEGLEN,STOLEN,TOL,WRKLEN
0043      1 !      INTEGER*4 S251,TWO15
0044      1 !      REAL*4      DEG,ERAD,PI,RAD
0045      INCLUDE 'CBC1.INC'
0046      1 ! -----CBC1.INC-----
0047      1 ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0048      1 ! -----
0049      1 ! BCOORD (-12:12,2) ?
0050      1 !
0051      1      INTEGER*2 BCOORD(-12:12,2)
0052      1
0053      1      COMMON /CBC/ BCOORD
0054      1 ! -----END CBC1.INC-----
0055      1
0056      INCLUDE 'CS.INC'
0057      1 ! -----CS-----
0058      1 ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0059      1 ! -----

```

```

0060 1 ! S      -1,3800    POLYGON AND SEGMENT STORAGE ARRAY      REAL*4
0061 1 ! STOLEN          STORAGE ARRAY LENGTH (FOR SEGS & POLYS) PARM
0062 1 !
0063 1          REAL*4  S(-1:STOLEN)
0064 1
0065 1          COMMON /CS/      S
0066 1 ! -----CS-END-----
0067 1
0068 !
0069 ! VARBL  SIZE      PURPOSE                                TYPE      RANGE
0070 ! -----
0071 ! DDNEG          DISTANCE TO FURTHEST JJ-J CROSS      INTEGER*2
0072 ! DDPOS          OVER POINT IN NEG,POS DIRECTION    INTEGER*2
0073 ! DNEG          DISTANCE TO FURTHEST J-JJ CROSS      INTEGER*2
0074 ! DPOS          OVER POINT IN NEG,POS DIRECTION    INTEGER*2
0075 ! DUM          GOSUB SIMULATION LABEL                INTEGER*2
0076 ! END1          FUNCTION                            INTEGER*2
0077 ! END2          FUNCTION                            INTEGER*2
0078 ! F            1ST ENDPOINT FOUND FLAG              BYTE
0079 ! FNP          FUNCTION                            INTEGER*2
0080 ! I            COUNTER                            INTEGER*2
0081 ! J            J-JJ INDEX                          INTEGER*2
0082 ! JJ           JJ-J INDEX                          INTEGER*2
0083 ! K            COUNTER                            INTEGER*2
0084 ! KK           COUNTER                            INTEGER*2
0085 ! KMOD         FUNCTION                            INTEGER*2
0086 ! N            CURRENT POLYGON                      INTEGER*2
0087 ! NEGT         OFFEST OF SEGMENT T                  INTEGER*2
0088 ! NUM          NUMBER OF POLYGON SEGMENTS            INTEGER*2
0089 ! PNDX (0,160) POLYGON INDEX ARRAY                  INTEGER*2
0090 ! PP1          SEGMENT JJ ENDPOINT                  INTEGER*2
0091 ! PP2          SEGMENT JJ ENDPOINT                  INTEGER*2
0092 ! P1           SEGMENT J ENDPOINT                   INTEGER*2
0093 ! P2           SEGMENT J ENDPOINT                   INTEGER*2
0094 ! R            ERROR RETURN FLAG                    BYTE
0095 ! SEG          1 OF 4 INTERIOR BORDER SEGMENTS      INTEGER*2  2,5,8,11
0096 ! T            SEGMENT                              INTEGER*2
0097 ! TT          SECOND ENDPOINT OF SEGMENT T          INTEGER*2
0098 ! X            LONGITUDE                            INTEGER*2
0099 ! Y            LATITUDE                             INTEGER*2
0100 !
0101 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0102
0103          INTEGER*4 TT
0104          INTEGER*2 DDNEG,DDPOS,DNEG,DPOS,DUM,I,J,JJ,K,KK,N,NEGT,NUM
0105          INTEGER*2 PNDX(0:4*POLLEN),P1,P2,PP1,PP2,SEG,T,X,Y
0106          INTEGER*2 END1,END2,FNP,KMOD      ! FUNCTIONS
0107          BYTE      R,F
0108
0109          R=.FALSE.          ! SET ERROR FLAG TO FALSE
0110          I=1                ! INITIALIZE LOOP I INDEX
0111 ! -----LOOP I-----
0112 1          IF (I.GT.PNDX(0)) GOTO 16          ! TEST FOR END OF LOOP I
0113          IF (PNDX(I).LE.0) GOTO 15          ! IF INPROPER POL, INCREMENT
0114 2          N=PNDX(I)          ! SET CURRENT POLYGON
0115          NUM=S(N+1)          ! SET NUMBER OF POL SEGMENTS
0116          J=1                ! INITIALIZE LOOP J INDEX
0117 ! -----LOOP J-----
0118

```

```

0119      3      IF (J.GT.S(N+1)-1) GOTO 15      ! TEST FOR END OF LOOP J
0120      IF (ABS(S(N+J+2)).NE.SEG) GOTO 14 ! IF NOT INTERIOR SEG
0121      F=.FALSE.      ! INIT 1ST ENDPT FOUND FLAG
0122      JJ=J+1      ! INITIALIZE LOOP JJ INDEX
0123
0124      !-----LOOP JJ-----
0125      4      IF (JJ.GT.NUM) GOTO 14      ! TEST FOR END OF LOOP JJ
0126      IF (S(N+JJ+2).NE.-S(N+J+2)) GOTO 13 ! SEGS OPPO DIRECTIONS
0127      IF (.NOT.F) THEN      ! NOT 1ST ENDPOINT FOUND
0128          P1=END1(N,J)      ! GET SEG J ENDPOINTS
0129          P2=END2(N,J)      ! GET SEG J ENDPOINTS
0130          F=.TRUE.      ! 1ST ENDPOINT FOUND FLAG
0131      END IF      ! END IF BLOCK
0132      PP1=END1(N,JJ)      ! GET REST OF SEGS ENDPTS (JJ)
0133      PP2=END2(N,JJ)      ! GET REST OF SEGS ENDPTS (JJ)
0134      ! ENDPTS ARE NOT CLOSE ENOUGH
0135      IF(ABS(P1-PP2).GT.TOL .OR. ABS(P2-PP1).GT.TOL) GOTO 13
0136
0137      !-----ELIMINATE DOUGHNUT MIDDLE SEGMENTS
0138
0139      !-----FIND MAX J-JJ CROSSOVER DISTANCE
0140      DPOS=-1      ! J-JJ CROSS OVER DISTANCE; >0
0141      DNEG=-1      ! J-JJ CROSS OVER DISTANCE; <0
0142      K=J      ! INITIALIZE INDEX K
0143      6      K=K+1      ! INCREMENT COUNTER K
0144      IF (K.LE.JJ-2) THEN      ! IF ALL J THRU JJ SEGS DONE
0145          ASSIGN 101 TO DUM      ! FIND OFFSET LAT OF LNG OF
0146          GOTO 200      ! SEG K VIA GOSUB SIMULATION
0147      101      IF (.NOT.F) GOTO 6      ! IF ENDPT NOT WITHIN LIMITS
0148      IF (SEG.LE.6) T=Y-P2      ! VERTICAL INTERIOR BORDER SEG
0149      IF (SEG.GT.6) T=X-P2      ! HORIZ INTERIOR BORDER SEG
0150      IF (T.GE.0) DPOS=MAX(DPOS,T) ! MAX ABSOLUTE VALUE OF
0151      IF (T.LT.0) DNEG=MAX(DNEG,-T)! CROSSOVER PT DISTANCES
0152      GOTO 6      ! DO NEXT SEGMENT
0153      END IF      ! END IF BLOCK
0154
0155      !-----FIND MAX JJ-J CROSSOVER DISTANCE
0156      DDPOS=-1      ! INIT JJ-J CROSSOVER POS DIST
0157      DDNEG=-1      ! INIT JJ-J CROSSOVER NEG DIST
0158      KK=JJ      ! INITIALIZE INDEX KK
0159      8      KK=KK+1      ! INCREMENT INDEX KK
0160      IF (KK.LE.J+NUM-2) THEN      ! IF ALL JJ THRU J SEGS DONE
0161          K=KMOD(KK-1,NUM)+1
0162          ASSIGN 102 TO DUM      ! FIND OFFSET LAT OF LNG OF
0163          GOTO 200      ! SEG K VIA GOSUB SIMULATION
0164      102      IF (.NOT.F) GOTO 8      ! IF ENDPT NOT WITHIN LIMITS
0165      IF (SEG.LE.6) T=Y-PP2      ! VERTICAL INTERIOR BORDER SEG
0166      IF (SEG.GT.6) T=X-PP2      ! HORIZ INTERIOR BORDER SEG
0167      IF (T.GE.0) DDPOS=MAX(DDPOS,T) ! MAX ABSOLUTE VALUE OF
0168      IF (T.LT.0) DDNEG=MAX(DDNEG,-T)! CROSSOVER PT DISTANCES
0169      GOTO 8      ! DO NEXT SEGMENT
0170      END IF      ! END IF BLOCK
0171
0172      !-----THE STRUCTURE IS INVALID-----
0173      IF(DPOS.LT.0.AND.DNEG.LT.0) GOTO 1001! NO CROSSPT FOR J-JJ
0174      IF(DDPOS.LT.0.AND.DDNEG.LT.0) GOTO 1001! NO JJ-J CROSSPT
0175      IF((DPOS.LT.0.OR.DDPOS.LT.0).AND.!(J-JJ & JJ-J CROSSPTS NOT
0176      *      (DNEG.LT.0.OR.DDNEG.LT.0)) GOTO 1001! ON THE SAME SIDE
0177      IF((DPOS.EQ.DDPOS .AND. DPOS.GE.0) .OR. ! IDENTICAL CROSS-

```

```

0178      *      (DNEG.EQ.DDNEG .AND. DNEG.GE.0)) GOTO 1001 ! POINTS
0179      IF(DPOS.LT.DDPOS.AND.DNEG.GT.DDNEG) GOTO 1001! CROSS OTHER
0180
0181      !-----WHICH POLYGON IS INTERIOR TO OTHER
0182      F=(DPOS.GT.DDPOS) ! ASSUME CROSSPTS ON POS SIDE
0183      ! IF J-JJ OR JJ-J DO NOT CROSS POS SIDE,
0184      IF (DPOS.LT.0 .OR. DDPOS.LT.0) F=(DNEG.GT.DDNEG) ! SET NEG
0185      DO 10 K=1,NUM ! CLEAN OUT POLYGON INTERIOR
0186      IF (.NOT. F.AND.K.GE.J.AND.K.LE.JJ) S(N+K+2)=0 ! SET=0
0187      IF (F .AND.(K.LE.J .OR. K.GE.JJ)) S(N+K+2)=0 ! SET=0
0188      10      CONTINUE ! END LOOP
0189
0190      !-----CHECK FOR & DELETE ADJACENT BORDERS
0191      IF (F) J=J+1 ! INCREASE J-JJ INDEX
0192      IF (.NOT.F) J=J-1 ! DECREASE J-JJ INDEX
0193      IF (J.LT.1) J=NUM ! SET TO NUM
0194      IF (.NOT.F) JJ=JJ+1 ! INCREASE JJ-J INDEX
0195      IF (F) JJ=JJ-1 ! DECREASE JJ-J INDEX
0196      IF (JJ.GT.NUM) JJ=1 ! SET TO 1
0197      IF (S(N+J+2).EQ.-S(N+JJ+2)) S(N+J+2)=0 ! DELETE ADJACENT
0198      IF (S(N+J+2).EQ.0) S(N+JJ+2)=0 ! BORDER SEGMENTS
0199      KK=0 ! RESET DELETED SEGMENT COUNT
0200
0201      !-----COMPRESS ARRAY-----
0202      DO 11 K=N+3,N+NUM+2 ! COMPRESS SEG REFERENCE LIST
0203      IF (S(K).NE.0) S(K-KK)=S(K) ! MOVE UP SEGMENT REFERENCES
0204      IF (S(K).EQ.0) KK=KK+1 ! RESET INDEX
0205      11      CONTINUE ! END DO LOOP
0206      DO 12 K=N+NUM+3,N+NUM+S(N+2)+2 ! COMPENSATE FOR COMPRESS
0207      IF (S(N+2).LE.0) GOTO 12 ! NOT LABEL POINTS IN POLYGON
0208      S(K-KK)=S(K) ! MOVE UP LABEL POINTS
0209      12      CONTINUE ! END DO LOOP
0210      S(N+1)=NUM-KK ! UPDATE SEGMENT COUNT
0211      GOTO 2 ! DO FOR NEXT POLYGON
0212      13      JJ=JJ+1 ! INCREMENT JJ INDEX
0213      GOTO 4 ! DO FOR NEXT J
0214
0215      !-----END OF LOOP JJ-----
0216      14      J=J+1 ! INCREMENT J INDEX
0217      GOTO 3 ! DO FOR NEXT JJ
0218
0219      !-----END OF LOOP J-----
0220      15      I=I+1 ! INCREMENT I INDEX
0221      GOTO 1 ! DO FOR NEXT I
0222
0223      !-----END OF LOOP I-----
0224      16      GO TO 999 ! RETURN TO CALLING ROUTINE
0225
0226      !=====GOSUB SIMULATION=====
0227      200      T=S(N+K+2) ! SET T TO SEG REF K IN POL N
0228      IF (ABS(T).LE.6) THEN ! VERTICAL INTERIOR BORDER SEG
0229      Y=END2(N,K) ! LAT OF 2ND ENDPT OF K
0230      NEGT=-T ! OFFSET OF SEMENT T
0231      X=BCOORD(NEGT,2) ! LNG OF BORDER COORDINATE
0232      END IF ! END IF BLOCK
0233      IF (ABS(T).GT.6.AND.ABS(T).LE.12) THEN ! HORIZ INTER BORDER SEG
0234      NEGT=-T ! OFFSET OF SEGMENT T
0235      Y=BCOORD(NEGT,1) ! LNG OF BORDER COORD OF T
0236      X=END2(N,K) ! LNG OF 2ND ENDPT OF K

```



```

0237         END IF                                ! END IF BLOCK
0238     IF (ABS(T).GT.12) THEN                        ! DIGITIZED SEGMENT
0239         TT=S(FNP(-T))                            ! LAST POINT IN SEGMENT T
0240         IF (TT.LT.0) GOTO 1001                    ! INVALID
0241         Y=TT/501                                  ! LATITUDE OFFSET OF TT
0242         X=JMOD(TT,501)                            ! INTEGER*4 MODULO; LNG OFFSET
0243         END IF                                    ! END IF
0244         IF (SEG.LE.6) F=(ABS(X-BCOORD(SEG,2)).LE.TOL) ! SET FLAG F
0245         IF (SEG.GT.6) F=(ABS(Y-BCOORD(SEG,1)).LE.TOL) ! SET FLAG F
0246         GOTO DUM
0247     !=====END GOSUB=====
0248
0249     !-----ERROR-----
0250 1001     WRITE(5,1002)                            ! INVALID POLYGON STRUCTURE
0251         R=.TRUE.                                ! SET ERROR FLAG TO TRUE
0252 999     RETURN                                    ! RETURN TO CALLING ROUTINE
0253
0254     !-----FORMAT STATEMENT-----
0255 1002     FORMAT(X,'INVALID POLYGON STRUCTURE ENCOUNTERED IN DNUT')
0256         END

```

```

0001      SUBROUTINE DRAW(IXRAST,IYRAST)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: DRAW
0005      ! AUTHOR: J. CASCIO, W. WACHTER(FORTRAN 77), NUSC/NL, CODE 3333
0006      ! DATE: 1981 & 9/84 (FORTRAN 77)
0007      ! FUNCTION: DRAW A VECTOR FROM THE CURRENT POSITION KBEAMX,KBEAMY TO
0008      !             IXRAST,IYRAST OBSERVING CLIPPING
0009      ! INPUTS: LOCATION FOR VECTOR
0010      ! OUTPUTS: VECTOR DRAWN
0011      ! MODULES CALLED: CLIPOS, CLIPT
0012      ! CALLED BY: AXIS, BOX, DRAWU, GRID, SVPGRF
0013      ! NOTE: THE VECTOR MAY BE MOVED BEYOND THE CLIPPED BOUNDARIES EVEN THOUGH
0014      !       IT MAY NOT BE DRAWN TO THAT POINT
0015      !
0016      INCLUDE 'SCREEN.INC'
0017      1 !-----SCREEN-----
0018      1 ! VARBL   SIZE   PURPOSE                                TYPE      RANGE
0019      1 ! -----
0020      1 ! ICLIP   (4)    CLIP BOUNDARIES                        INTEGER*2
0021      1 ! ISCLIP                CLIPPING FLAG                  INTEGER*2  TRUE FALSE
0022      1 ! LENX                LENGTH OF X GRAPHICS BOUNDARY    INTEGER*2
0023      1 ! LENY                LENGTH OF Y GRAPHICS BOUNDARY    INTEGER*2
0024      1 ! MAXX                MAXIMUM X GRAPHICS BOUNDARY      INTEGER*2
0025      1 ! MAXY                MAXIMUM Y GRAPHICS BOUNDARY      INTEGER*2
0026      1 ! MINX                MINIMUM X GRAPHICS BOUNDARY      INTEGER*2
0027      1 ! MINY                MINIMUM Y GRAPHICS BOUNDARY      INTEGER*2
0028      1
0029      1      INTEGER*2 ICLIP,LENX,LENY
0030      1      INTEGER*2 MAXX,MAXY,MINX,MINY
0031      1      INTEGER*2 ISCLIP
0032      1
0033      1      COMMON /SCREEN/MINX,MAXX,MINY,MAXY,LENX,LENY,ICLIP(4),ISCLIP
0034      1 !-----SCREEN END-----
0035      INCLUDE 'TK4025.INC'
0036      1 !-----TK4025-----
0037      1 ! VARBL   SIZE   PURPOSE                                TYPE      RANGE
0038      1 ! -----
0039      1 ! KBEAMX                CURRENT BEAM X POSITION          INTEGER*2
0040      1 ! KBEAMY                CURRENT BEAM Y POSITION          INTEGER*2
0041      1
0042      1      INTEGER*2 KBEAMX,KBEAMY
0043      1
0044      1      COMMON/TK4025/KBEAMX,KBEAMY
0045      1 !-----TK4025 END-----
0046      !
0047      ! VARBL   SIZE   PURPOSE                                TYPE      RANGE
0048      ! -----
0049      ! CLIP                CLIP FLAG                          LOGICAL*2
0050      ! I                    PREVIOUS COORDINATE OUTSIDE CLIP FLAG INTEGER*2
0051      ! IX1                  STARTING X COORDINATE              INTEGER*2
0052      ! IX2                  ENDING X COORDINATE                INTEGER*2
0053      ! IXRAST              ENDING X COORDINATE IN RASTERS      INTEGER*2
0054      ! IY1                  STARTING Y COORDINATE              INTEGER*2
0055      ! IY2                  ENDING Y COORDINATE                INTEGER*2
0056      ! IYRAST              ENDING Y COORDINATE IN RASTERS      INTEGER*2
0057      ! J                    CURRENT COORDINATE OUTSIDE CLIP FLAG INTEGER*2
0058      ! K                    OUTSIDE CLIP AREA FLAG            INTEGER*2
0059      ! L                    COMPARE CLIP AREA FLAG            INTEGER*2

```

```

0060 ! L1      (4)  STORE PREVIOUS PTS OUT OF CLIP AREA   LOGICAL*2
0061 ! L2      (4)  STORE CURRENT POINTS OUT OF CLIP AREA LOGICAL*2
0062 !
0063      INTEGER*2 I,IX1,IX2,IXRAST,IY1,IY2,IYRAST,J,K,L
0064      LOGICAL*2 CLIP,L1(4),L2(4)
0065
0066 !-----SET UP WHERE THE DRAW VECTOR IS
0067      IX1=KBEAMX      ! STARTING X COORDINATE
0068      IY1=KBEAMY      ! STARTING Y COORDINATE
0069      IX2=IXRAST      ! ENDING X COORDINATE
0070      IY2=IYRAST      ! ENDING Y COORDINATE
0071      IF(IX1.EQ.IXRAST.AND.IY1.EQ.IYRAST) GO TO 999 ! SAME POINT, RETU
0072      CLIP=.FALSE.    ! INITIALIZE CLIP FLAG
0073      IF (.NOT. ISCLIP) GOTO 6      ! NO CLIP NEEDED
0074      CALL CLIPOS(IX1,IY1,I,L1)    ! CHECK WHERE PREVIOUS IS OUT
0075      CALL CLIPOS(IX2,IY2,J,L2)    ! CHECK WHERE CURRENT IS OUT
0076      IF (J.NE.0) CLIP=.TRUE.      ! SET FLAG FOR AT LEAST SECOND M
0077      IF(I.NE.0.AND.J.NE.0) GOTO 2  ! BOTH OUTSIDE CLIP AREA
0078      IF(I.EQ.0.AND.J.EQ.0) GOTO 6  ! BOTH INSIDE CLIP AREA
0079      I=IIABS(I)      ! PREVIOUS COORDINATE OUTSIDE CL
0080      J=IIABS(J)      ! CURRENT COORDINATE OUTSIDE CLI
0081      CALL CLIPT(IX1,IY1,IX2,IY2,I,J) ! CLIP LINES
0082      GOTO 6          ! BOTH POINTS ARE IN
0083
0084 !-----POINTS ARE BOTH OUT-----
0085 2      IF(IIABS(I).EQ.IIABS(J)) GOTO 8 ! IF POINTS IN SAME QUADRANT THE
0086      IF (I/3+J/3.EQ.-1) GOTO 5      ! IF PTS IN THE SAME CORNER THEN
0087      L=0                            ! INITIALIZE
0088      DO 3 K=1,4                    ! DO FOUR TIMES
0089          L=(L1(K) .AND. L2(K))+L    ! CLIPPED AREAS WHERE PTS ARE OU
0090 3      CONTINUE                    ! END DO LOOP
0091      IF (L.EQ.-1) GOTO 8            ! NO CLIP NEEDED
0092      IF (L.EQ.0) GOTO 5            ! BOTH ENDS NEED TO BE CLIPPED
0093      WRITE(5,4)                    ! ERROR
0094      GO TO 999                     ! RETURN TO CALLING ROUTINE
0095
0096 !-----CLIP BOTH ENDS OF THE VECTOR-----
0097 5      I=IIABS(I)                  ! CURRENT COORDINATE OUTSIDE CLI
0098      J=IIABS(J)                  ! CURRENT COORDINATE OUTSIDE CLI
0099      CALL CLIPT(IX1,IY1,IX2,IY2,I,J) ! GET NEW ENDPOINTS WITHIN CLIP
0100      IF (I/3+J/3.NE.1) GOTO 6      ! CHECK THE CORNER
0101      CALL CLIPOS(IX1,IY1,K,L1)      ! CHECK IF OUTSIDE CLIP AREA
0102      IF (K.NE.0) GOTO 8            ! IF NEW POINT IS OUT THEN MOVE
0103 6      WRITE(5,7) IX1,IY1,IX2,IY2  ! DRAW (CALCULATED) VECTOR
0104 8      KBEAMX=IXRAST              ! UPDATE CURRENT POSITION
0105      KBEAMY=IYRAST              ! UPDATE CURRENT POSITION
0106  C      IF (CLIP) WRITE(5,9) KBEAMX,KBEAMY ! COMMENTED OUT
0107 999      RETURN                    ! RETURN TO CALLNG ROUTINE
0108
0109 !-----FORMAT STATEMENTS-----
0110 4      FORMAT(' ** ERROR IN ''DRAW'', IMPOSSIBLE CLIP BOUNDARIES')
0111 7      FORMAT(' !VEC',4(X,I3))
0112 9      FORMAT(' !VEC ',I3,X,I3)
0113      END

```

```

0001      SUBROUTINE DUPDEP(NBT,D,VEL)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: DUPDEP
0005      ! AUTHOR: S. LAFLEUR & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1983 & 11/83 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE DUPDEP ELIMINATES DUPLICATE CONSECUTIVE DEPTHS
0008      !               AND DEPTHS WHICH ARE NOT IN ASCENDING ORDER.
0009      ! INPUTS:  PARAMETERS PASSED IN.
0010      ! OUTPUTS: MODIFIED PARAMETERS PASSED OUT.
0011      ! MODULES CALLED: NONE
0012      ! CALLED BY: ENVIRN, FORCST, XBT
0013      !
0014      !  VARBL   SIZE      PURPOSE                                TYPE      RANGE
0015      !  -----
0016      !  D       (25)      DEPTH                                REAL*4
0017      !  DLYR                                LAYER DEPTH          REAL*4
0018      !  I                                COUNTER              INTEGER*2
0019      !  J                                COUNTER              INTEGER*2
0020      !  NBT                                NUMBER OF BT POINTS  INTEGER*2
0021      !  NDLYR                               POSITON OF BT LAYER  INTEGER*2
0022      !  VEL       (25)      VELOCITY                            REAL*4
0023      !
0024      INTEGER*2 I,J,NBT,NDLYR
0025      REAL*4    D,DLYR,VEL
0026      DIMENSION D(1),VEL(1)
0027
0028      I = 1                                ! INITIALIZE COUNTER
0029      10  IF(I.LE.NBT-1) THEN                ! IF < NEXT TO LAST BT
0030          IF(D(I).GE.D(I+1)) THEN            ! NOT IN ASCENDING ORDER
0031              DO 50 J = I+1,NBT-1          ! ELIMINATE POINT AT I+1 BY
0032                  D(J) = D(J+1)              ! MOVING DEPTH VALUES UP ONE
0033                  VEL(J) = VEL(J+1)          ! & MOVING VEL VALUES UP ONE
0034      50  CONTINUE                          ! END DO LOOP
0035          I = I - 1                          ! DECREASE COUNTER
0036          NBT = NBT - 1                      ! DECREASE NUMBER OF BTS
0037          END IF                             ! END IF BLOCK
0038          I = I + 1                          ! INCREASE COUNTER
0039          GOTO 10                            ! START AGAIN
0040      . END IF                              ! END IF BLOCK
0041
0042      RETURN                                ! RETURN TO CALLING ROUTINE
0043      END                                  ! END SUBROUTINE

```

```

0001          SUBROUTINE DUPVEL(NBT,D,VEL)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: DUPVEL
0005      ! AUTHOR: S. LAFLEUR & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 5/84 & 5/84 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE DUPVEL ELIMINATES DUPLICATE CONSECUTIVE
0008      !              SOUND SPEEDS
0009      ! INPUTS:  PARAMETERS PASSED IN.
0010      ! OUTPUTS: MODIFIED PARAMETERS PASSED OUT.
0011      ! MODULES CALLED: NONE
0012      ! CALLED BY: ENVIRN,FORCST,XBT
0013      !
0014      ! VARBL  SIZE      PURPOSE                      TYPE      RANGE
0015      ! -----
0016      ! D      (25)      DEPTH                          REAL*4
0017      ! IDONE                      FLAG                  INTEGER*2
0018      ! J                      COUNTER                  INTEGER*2
0019      ! NBT                      NUMBER OF BT POINTS     INTEGER*2
0020      ! VEL      (25)      VELOCITY                      REAL*4
0021      !
0022      INTEGER*2 IDONE,J,NBT
0023      REAL*4    D,VEL
0024      DIMENSION D(1),VEL(1)
0025
0026      10      IDONE = 1                                ! INITIALIZE FLAG
0027
0028      DO 50 J=2,NBT                                    ! FOR NUMBER OF BT
0029      IF(VEL(J).EQ.VEL(J-1)) IDONE=0 ! RESET FLAG
0030      IF(VEL(J).EQ.VEL(J-1)) VEL(J)=VEL(J)+0.01 ! RESET VEL
0031      50      CONTINUE                                ! END DO LOOP
0032
0033      IF(IDONE.EQ.0) GOTO 10                            ! REPEAT
0034
0035      RETURN                                           ! RETURN TO CALLING ROUTINE
0036      END                                             ! END SUBROUTINE

```

```

0001      SUBROUTINE EDITBT(INSSP,NBT,D,TVEL)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: EDITBT
0005      ! AUTHOR: G. BROWN & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1974 & 12/83 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE EDITBT ALLOWS THE OPERATOR EDIT BT DATA.
0008      ! INPUTS: OPERATOR SELECTION TO UPDATE PARAMETERS OR NOT.
0009      !          VARIABLES PASSED IN.
0010      ! OUTPUTS: CRT PROMPTING MESSAGES TO OPERATOR.
0011      ! MODULES CALLED: ICLR
0012      ! CALLED BY: BT,KEYPCH
0013      !
0014      ! VARBL   SIZE      PURPOSE                                TYPE      RANGE
0015      ! -----
0016      ! D              DEPTH MATRIX                             REAL*4
0017      ! I              COUNTER                                  INTEGER*2
0018      ! INSSP          INPUTED SSP                               INTEGER*2
0019      ! J              COUNTER                                  INTEGER*2
0020      ! K              COUNTER                                  INTEGER*2
0021      ! L1            LINE NUMBER FOR CHANGE                   INTEGER*2
0022      ! M              NUMBER OF BT + 1                         INTEGER*2
0023      ! NBT            NUMBER OF DEPTH/TEMP PAIRS               INTEGER*2
0024      ! N4             TYPE OF CORRECTION                       INTEGER*2      1,2,3
0025      ! TVEL          TEMPERATURE MATRIX                       REAL*4
0026
0027      INTEGER*2 I,INSSP,J,K,L1,M,NBT,N4
0028      REAL*4    D,TVEL
0029      DIMENSION D(1),TVEL(1)
0030
0031      !-----PRELIMINARIES-----
0032      10  CALL ICLR                      ! CLEAR SCREEN
0033          WRITE(5,1200)                  ! WRITE TITLES
0034          DO 15 I=1,NBT                  ! FOR ALL BT POINTS
0035              WRITE(5,1230) I,D(I),TVEL(I) ! WRITE DEPTH, TEMP, SS
0036          15  CONTINUE                  ! END DO LOOP
0037              M=NBT+1                    ! M IS NUMBER OF BT + 1
0038
0039      !-----CORRECTION-----
0040          WRITE(5,1010) M                ! INPUT LINE NUMBER FOR CHANGE
0041          WRITE(5,1020)                  ! INPUT LINE NUMBER FOR CHANGE
0042          READ(5,1050) L1                ! LINE NUMBER FOR CHANGE
0043          IF(L1.EQ.0) GO TO 370          ! NO CHANGES WANTED, RETURN
0044          IF(L1.LT.1.OR.L1.GT.M) GO TO 10 ! INVALID LINE #, TRY AGAIN
0045      20  WRITE(5,1100)                  ! INPUT TYPE OF CORRECTION
0046          READ(5,1150) N4                ! TYPE OF CORRECTION
0047          IF(N4.LT.1.OR.N4.GT.3) GO TO 20 ! INVALID, ASK AGAIN
0048
0049          IF(N4.EQ.1) THEN                ! DELETE AN ENTRY
0050              IF(L1.NE.NBT) THEN          ! NOT LAST ENTRY
0051                  IF(L1.GT.NBT) GO TO 10  ! GREATER THAN LAST, TRY AGAIN
0052                  K=NBT-1                ! K IS NUMBER OF BT - 1
0053                  DO 130 J=L1,K          ! MOVE EACH ARRAY VALUE UP ONE
0054                      D(J)=D(J+1)         ! DEPTH
0055                      TVEL(J)=TVEL(J+1)  ! TEMPERATURE
0056      130  CONTINUE                      ! THIS DELETES ENTRY AT L1 BY
0057              END IF                    ! WRITING OVER IT
0058              D(NBT)=0.                 ! ZERO OUT LAST DEPTH ENTRY
0059              TVEL(NBT)=0.              ! ZERO OUT LAST TEMP ENTRY

```

```

0060      NBT=NBT-1      ! DECREASE # OF BT BY ONE
0061      END IF      ! END IF BLOCK
0062
0063      IF(N4.EQ.2) THEN      ! CHANGE AN ENTRY
0064          IF(L1.GT.NBT) GO TO 10      ! GREATER THAN LAST, TRY AGAIN
0065          WRITE(5,1300)      ! INPUT DEPTH
0066          READ(5,1330) D(L1)      ! REVISED DEPTH
0067          WRITE(5,1350)      ! INPUT TEMPERATURE
0068          READ(5,1330) TVEL(L1)      ! REVISED TEMPERATURE
0069          END IF      ! CHANGED BY WRITTING OVER OLD
0070
0071      IF(N4.EQ.3) THEN      ! ADD AN ENTRY
0072          IF(INSSP.NE.5.AND.NBT.GE.25) GO TO 10      ! > MAX, TRY AGAIN
0073          IF(INSSP.EQ.5.AND.NBT.GE.50) GO TO 10      ! > MAX, TRY AGAIN
0074          IF(L1.NE.M) THEN      ! NOT EQUAL TO LAST
0075              DO 330 K=M,L1,-1      ! MOVE ARRAY VALUES DOWN ONE
0076                  D(K)=D(K-1)      ! MOVE DEPTH VALUES
0077                  TVEL(K)=TVEL(K-1)      ! MOVE TEMP VALUES
0078          330      CONTINUE      ! MOVING VALUES DOWN ONE ALLOWS
0079              END IF      ! ROOM FOR NEW ENTRY AT L1
0080              WRITE(5,1300)      ! INPUT DEPTH
0081              READ(5,1330) D(L1)      ! REVISED DEPTH
0082              WRITE(5,1350)      ! INPUT TEMPERATURE
0083              READ(5,1330) TVEL(L1)      ! REVISED TEMPERATURE
0084              NBT=NBT+1      ! INCREASE # OF BT BY 1
0085              END IF      ! END IF BLOCK
0086          GO TO 10      ! START AGAIN
0087      370      RETURN      ! RETURN TO CALLING ROUTINE
0088
0089      !-----FORMAT STATEMENTS-----
0090      1010      FORMAT(//,T20,'****ENTER LINE NOS. 1 -,I2,' FOR CHANGES****'
0091          1      /1H ,T20,'****ENTER EXTRA (CR) FOR END OF EDIT****')
0092      1100      FORMAT(1H /T20,'TYPES OF CORRECTION'
0093          1      /1H ,T20,'1 = DELETE ENTRY'/1H ,T20,'2 = CHANGE ENTRY'
0094          2      /1H ,T20,'3 = INSERT NEW ENTRY BEFORE LINE INDICATED'
0095          3      //1H$,T24,'ENTER TYPE OF CORRECTION (X)',T65,' ')
0096      1020      FORMAT(1H /1H$,T24,'ENTER LINE NUMBER (XX)',T65,' ')
0097      1050      FORMAT(I2)
0098      1150      FORMAT(I1)
0099      1200      FORMAT(1H /T20,'NO.',T29,'DEPTH',T42,'TEMP'
0100          1      /T43,'OR'/T42,'SOUND'/T42,'SPEED'/)
0101      1230      FORMAT(T20,I2,2(5X,F7.1))
0102      1300      FORMAT(1H /1H$,T24,'ENTER DEPTH (ONE DECIMAL PLACE)',T65,' ')
0103      1330      FORMAT(F10.0)
0104      1350      FORMAT(1H /1H$,T24,'ENTER TEMP (ONE DECIMAL PLACE)',T65,' ')
0105      END

```

```

0001             INTEGER*2 FUNCTION END1(I,J)
0002
0003 ! PROLOGUE:
0004 ! MODULE NAME: END1
0005 ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006 ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007 ! DATE: 1982 & 6/84 (FORTRAN 77)
0008 ! FUNCTION: SUBROUTINE ENDPOINT #1 WILL DETERMINE EITHER THE
0009 !             DISTANCE OF THE X OR Y DISTANCE OF THE FIRST
0010 !             ENDPOINT OF SEGMENT J OF POLYGON I.
0011 ! INPUTS: VARIABLES NEEDED TO CALCULATE DISTANCE
0012 ! OUTPUTS: X OR Y DISTANCE OF THE FIRST ENDPOINT OF J
0013 ! MODULES CALLED: BMOD, FNP, KMOD
0014 ! CALLED BY: CNNCT, DNUT, PDIST
0015 !
0016             INCLUDE 'MAP.PAR'
0017 1             PARAMETER STOLEN=3800
0018 1             PARAMETER SEGLEN=60, POLLEN=40
0019 1             PARAMETER WRKLEN=1000, NDXLEN=300
0020 1             PARAMETER MAXDTY=3
0021 1             PARAMETER TOL=3
0022 1             PARAMETER DEG=57.2957795
0023 1             PARAMETER RAD=.017453293
0024 1             PARAMETER PI=3.14159265
0025 1             PARAMETER ERAD=3440.3
0026 1             PARAMETER S251=63001
0027 1             PARAMETER TWO15=32768
0028 1
0029 1 !             INTEGER*2 MAXDTY,NDXLEN,POLLEN,SEGLEN,STOLEN,TOL,WRKLEN
0030 1 !             INTEGER*4 S251,TWO15
0031 1 !             REAL*4     DEG,ERAD,PI,RAD
0032             INCLUDE 'CBC2.INC'
0033 1 ! -----CBC2.INC-----
0034 1 ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0035 1 ! -----
0036 1 ! BCOORD (25,2)
0037 1 !
0038 1             INTEGER*2 BCOORD(25,2)
0039 1
0040 1             COMMON /CBC/  BCOORD
0041 1 ! -----END CBC2.INC-----
0042 1
0043             INCLUDE 'CS.INC'
0044 1 ! -----CS-----
0045 1 ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0046 1 ! -----
0047 1 ! S      -1,3800    POLYGON AND SEGMENT STORAGE ARRAY      REAL*4
0048 1 ! STOLEN                STORAGE ARRAY LENGTH (FOR SEGS & POLYS) PARM
0049 1 !
0050 1             REAL*4  S(-1:STOLEN)
0051 1
0052 1             COMMON /CS/   S
0053 1 ! -----CS-END-----
0054 1
0055 !
0056 ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0057 ! -----
0058 ! BMOD                FUNCTION                                INTEGER*2
0059 ! FNP                  FUNCTION                                INTEGER*2

```



```

0060      ! I          CURRENT POLYGON          INTEGER*2
0061      ! J          CURRENT SEGMENT IN POLYGON I  INTEGER*2
0062      ! K          SEGMENT PRECEDING J          INTEGER*2
0063      ! KMOD       FUNCTION                  INTEGER*2
0064      !
0065      ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0066
0067          INTEGER*2 FNP,I,J,K,KMOD
0068          REAL*4  BMOD
0069
0070          END1=-1                                ! INITIALIZE ENDPOINT #1
0071          K=-S(I+(KMOD(J-2,IIFIX(S(I+1)))+1)+2)! SEGMENT PRECEDING J
0072          IF (ABS(K).LE.12) THEN                  ! INTERIOR BORDER SEGMENT
0073              IF (ABS(S(I+J+2)).LE.12.) END1=BCOORD(K+13,2) ! HORIZONTAL
0074              IF (ABS(S(I+J+2)).LE.6.) END1=BCOORD(K+13,1)  ! VERTICAL
0075          ELSE                                     ! BORDER SEGMENT
0076              IF (ABS(S(I+J+2)).LE.12.) END1=IIFIX(BMOD(S(FNP(K)),501))!HOR
0077              IF (ABS(S(I+J+2)).LE.6.) END1=IIFIX(S(FNP(K))/501) ! VERTICAL
0078          END IF                                  ! END IF BLOCK
0079          IF (END1.EQ.-1) WRITE(5,102)             ! ERROR MESSAGE
0080          RETURN                                   ! RETURN TO CALLING ROUTINE
0081
0082      !-----FORMAT STATEMENT-----
0083      102      FORMAT(X,'PROGRAM ERROR, ATTEMPT TO USE END1 ON NON-BORDER')
0084      END

```

#### COMMAND QUALIFIERS

```

FORTRAN /CHECK=ALL/LIST/SHOW=(INCLUDE,NOMAP) [LAFLEUR]END1.F77

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)
/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD_LINES /NOCROSS_REFERENCE

```

#### COMPILATION STATISTICS

```

Run Time:          1.32 seconds
Elapsed Time:      4.48 seconds
Page Faults:       351
Dynamic Memory:    135 pages

```

```

0001          INTEGER*2 FUNCTION END2(I,J)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: END2
0005      ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006      ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007      ! DATE: 1982 & 6/84 (FORTRAN 77)
0008      ! FUNCTION: SUBROUTINE ENDPOINT #2 WILL DETERMINE EITHER
0009      !              THE X OR Y DISTANCE OF THE LAST ENDPOINT OF
0010      !              SEGMENT J OF POLYGON I.
0011      ! INPUTS: VARIABLES NEEDED TO CALCULATE DISTANCE
0012      ! OUTPUTS: THE X OR Y DISTANCE OF THE LAST ENDPOINT OF J
0013      ! MODULES CALLED: BMOD, FNP, KMOD
0014      ! CALLED BY: CNNCT, DNUT, PDIST
0015      !
0016          INCLUDE 'MAP.PAR'
0017      1      PARAMETER STOLEN=3800
0018      1      PARAMETER SEGLEN=60, POLLEN=40
0019      1      PARAMETER WRKLEN=1000, NDXLEN=300
0020      1      PARAMETER MAXDTY=3
0021      1      PARAMETER TOL=3
0022      1      PARAMETER DEG=57.2957795
0023      1      PARAMETER RAD=.017453293
0024      1      PARAMETER PI=3.14159265
0025      1      PARAMETER ERAD=3440.3
0026      1      PARAMETER S251=63001
0027      1      PARAMETER TWO15=32768
0028      1
0029      1 !      INTEGER*2 MAXDTY, NDXLEN, POLLEN, SEGLEN, STOLEN, TOL, WRKLEN
0030      1 !      INTEGER*4 S251, TWO15
0031      1 !      REAL*4      DEG, ERAD, PI, RAD
0032          INCLUDE 'CBC2.INC'
0033      1 ! -----CBC2.INC-----
0034      1 ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0035      1 ! -----
0036      1 ! BCOORD (25,2)
0037      1 !
0038      1      INTEGER*2 BCOORD(25,2)
0039      1
0040      1      COMMON /CBC/      BCOORD
0041      1 ! -----END CBC2.INC-----
0042      1
0043          INCLUDE 'CS.INC'
0044      1 ! -----CS-----
0045      1 ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0046      1 ! -----
0047      1 ! S      -1,3800    POLYGON AND SEGMENT STORAGE ARRAY      REAL*4
0048      1 ! STOLEN          STORAGE ARRAY LENGTH (FOR SEGS & POLYS) PARM
0049      1 !
0050      1      REAL*4  S(-1:STOLEN)
0051      1
0052      1      COMMON /CS/      S
0053      1 ! -----CS-END-----
0054      1
0055      !
0056      ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0057      ! -----
0058      ! BMOD          FUNCTION                                REAL*4
0059      ! FNP           FUNCTION                                INTEGER*2

```

```

0060      ! I          CURRENT POLYGON          INTEGER*2
0061      ! J          CURRENT SEGMENT OR POLYGON I  INTEGER*2
0062      ! K          SEGMENT SUCCEEDING J        INTEGER*2
0063      ! KMOD       FUNCTION                  INTEGER*2
0064      !
0065      ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0066
0067      INTEGER*2 FNP,I,J,K,KMOD
0068      REAL*4  BMOD
0069
0070      END2=-1                                ! INITIALIZE ENDPOINT #2
0071      K=S(I+(KMOD(J,IIFIX(S(I+1)))+1)+2)! SEGMENT SUCCEEDING J
0072      IF (ABS(K).LE.12) THEN                ! INTERIOR BORDER SEGMENT
0073          IF (ABS(S(I+J+2)).LE.12.) END2=BCOORD(K+13,2) ! HORIZONTAL
0074          IF (ABS(S(I+J+2)).LE.6.) END2=BCOORD(K+13,1)  ! VERTICAL
0075      ELSE                                  ! BORDER SEGMENT
0076          IF (ABS(S(I+J+2)).LE.12.) END2=IIFIX(BMOD(S(FNP(K)),501))!HOR
0077          IF (ABS(S(I+J+2)).LE.6.) END2=IIFIX(S(FNP(K))/501) ! VERTICAL
0078      END IF
0079      IF (END2.EQ.-1) WRITE(5,102)           ! ERROR MESSAGE
0080      RETURN                                ! RETURN TO CALLING ROUTINE
0081
0082      !-----FORMAT STATEMENT-----
0083      102      FORMAT(X,'PROGRAM ERROR, ATTEMPT TO USE END2 ON NON-BORDER')
0084      END

```

#### COMMAND QUALIFIERS

```
FORTRAN /CHECK=ALL/LIST/SHOW=(INCLUDE,NOMAP) [LAFLEUR]END2.F77
```

```
/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
```

```
/DEBUG=(NOSYMBOLS,TRACEBACK)
```

```
/STANDARD=(NOSYNTAX,NOSOURCE_FORM)
```

```
/SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)
```

```
/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD_LINES /NOCROSS_REFERENCE
```

#### COMPILATION STATISTICS

```

Run Time:          1.23 seconds
Elapsed Time:      10.02 seconds
Page Faults:       349
Dynamic Memory:    135 pages

```

SUBROUTINE ENVIRN(IPRINT)

```

0001
0002
0003 ! PROLOGUE:
0004 ! MODULE NAME: ENVIRN
0005 ! AUTHOR: SUNG KO, S. LAFLEUR & W. WACHTER, CODE 3333, NUSC/NLL
0006 ! DATE: 1974, 1983 (REDESIGN), & 12/83 (FORTRAN 77)
0007 ! FUNCTION: SUBROUTINE ENVIRN IS USED TO OBTAIN LOCAL ENVIRONMENTAL
0008 ! PARAMETERS WHICH ARE COMBINED WITH HISTORICAL DATA FOR
0009 ! THE AREA AND MONTH.
0010 ! INPUTS: PARAMETERS PASSED IN. VARIABLES IN COMMONS.
0011 ! OUTPUTS: CRT PROMPTING MESSAGES TO OPERATOR.
0012 ! MODULES CALLED: BT,DUPDEF,DUPVEL,ICLR,INSERT,KEYFCH,LAYER,MAP,
0013 ! NOCONV,SVPGRF,VELTMP,XBT
0014 ! CALLED BY: SIMAS
0015 !
0016 ! INCLUDE 'DHST.INC'
0017 1 ! -----DHST-----
0018 1 ! VARBL SIZE PURPOSE TYPE RANGE
0019 1 ! -----
0020 1 ! SCHNLD SOUND CHANNEL LAYER DEPTH REAL*4
0021 1 !
0022 1 REAL*4 SCHNLD
0023 1
0024 1 COMMON /DHST/ SCHNLD
0025 1 ! -----DHST END-----
0026 ! INCLUDE 'DTV.INC'
0027 1 ! -----DTV-----
0028 1 ! VARBL SIZE PURPOSE TYPE RANGE
0029 1 ! -----
0030 1 ! D (25) DEPTH REAL*4
0031 1 ! DD (25) DEPTH REAL*4
0032 1 ! NNBT NUMBER OF BATHETHERMAL INTEGER*2
0033 1 ! T (25) TEMPERATURE REAL*4
0034 1 ! TT (25) TEMPERATURE REAL*4
0035 1 ! VEL (25) VELOCITY REAL*4
0036 1 !
0037 1 INTEGER*2 NNBT
0038 1 REAL*4 D,DD,T,TT,VEL
0039 1
0040 1 COMMON /DTV/ D(25),T(25),VEL(25),DD(25),TT(25),NNBT
0041 1 ! -----END DTV-----
0042 ! INCLUDE 'ENVN.INC'
0043 1 ! -----ENVN-----
0044 1 ! VARBL SIZE PURPOSE TYPE RANGE
0045 1 ! -----
0046 1 ! BIO (2) BIOLOGICAL BACK SCATTERING REAL*4 -57. & -47.
0047 1 ! DLVR LAYER DEPTH REAL*4
0048 1 ! MGS MGS PROVINCE INTEGER*2
0049 1
0050 1 REAL*4 BIO,DLVR
0051 1 INTEGER*2 MGS
0052 1 DATA BIO/-57.,-47./
0053 1
0054 1 COMMON /ENVN/ BIO(2),DLVR,MGS
0055 1
0056 1 ! -----END ENVN-----
0057 ! INCLUDE 'GRF.INC'

```

```

0058 1 ! -----GRF-----
0059 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0060 1 ! -----
0061 1 ! DBT      (25)  DEPTH OF DEPTH/VEL PAIR                REAL*4
0062 1 ! IANS      PREDICTION TYPE                                INTEGER*2  -2 TO +2
0063 1 ! ILYR      INDEX FOR LAYER DEPTH                            INTEGER*2
0064 1 ! INBT      OPERATOR ENTERED # OF BT POINTS              INTEGER*2
0065 1 ! ISVP      LATEST OR HISTORICAL BT FLAG                  INTEGER*2  1 OR 2
0066 1 ! I2000    SVP INDEX FOR 2000 FT DEPTH                  INTEGER*2
0067 1 ! VBT      (25)  VELOCITY FOR DEPTH PAIR REAL*4          REAL*4
0068 1
0069 1          REAL*4  DBT,VBT
0070 1          INTEGER*2 IANS,ILYR,INBT,ISVP,I2000
0071 1
0072 1          COMMON /GRF/ IANS,ISVP,ILYR,I2000,INBT,DBT(25),VBT(25)
0073 1
0074 1 ! -----END GRF-----
0075          INCLUDE 'LOC.INC'
0076 1 ! -----LOC-----
0077 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0078 1 ! -----
0079 1 ! INDX      SSP INDEX                                INTEGER*2
0080 1 ! LAT      (4)  LATITUDE                                INTEGER*2
0081 1 ! LONG     (4)  LONGITUDE                                INTEGER*2
0082 1 ! NMAREA   (20) AREA OCEAN NAME                          BYTE
0083 1 ! NOC      NUMBER OF OCEAN                            INTEGER*2
0084 1 ! RCZ      RANGE TO CONVERG. ZONE REAL*4
0085 1
0086 1          REAL*4  RCZ
0087 1          INTEGER*2 INDX,LAT,LONG,NOC
0088 1          BYTE    NMAREA(20)
0089 1
0090 1          COMMON /LOC/ LAT(4),LONG(4),NOC,INDX,RCZ,NMAREA
0091 1
0092 1 ! -----END LOC-----
0093          INCLUDE 'SVP.INC'
0094 1 ! -----SVP-----
0095 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0096 1 ! -----
0097 1 ! BDF      BOTTOM DEPTH IN FATHOMS                        REAL*4
0098 1 ! BIOP      BIOLOGICAL BACK SCATTERING COEF              REAL*4
0099 1 ! BTDATE   (9)  DATE OF LAST BT INPUT                    BYTE
0100 1 ! BTTIME   (8)  TIME OF LAST BT INPUT                    BYTE
0101 1 ! C        (50)  VELOCITY (PAIRED WITH Z FOR SVP)        REAL*4
0102 1 ! CC       (50)  VELOCITY (PAIRED WITH ZZ FOR SVP)       REAL*4
0103 1 ! CS      SOUND VELOCITY AT SURFACE                      REAL*4
0104 1 ! DEG      TEMPERATURE (DEG)                            REAL*4  57.2957795
0105 1 ! EL      LAYER DEPTH                                    DATA
0106 1 ! F        FREQUENCY                                    REAL*4
0107 1 ! GRDS     GRIDS                                        REAL*4  0.0164
0108 1 ! ITO      MINIMAL 2-WAY TRAVEL TIME                    INTEGER*2
0109 1 ! MGSOP     MGS PROVINCE NUMBER                          INTEGER*2
0110 1 ! N        # OF DEPTH/VELOCITY PAIRS                     INTEGER*2
0111 1 ! NN       # OF DEPTH/VELOCITY PAIRS                     INTEGER*2
0112 1 ! PI      MATHEMATICAL CONSTANT PI                      REAL*4  3.1415927
0113 1 ! SNDATE   (9)  DATE SYS PARMS LAST UPDATED              BYTE
0114 1 ! SNTIME   (8)  TIME SYS PARMS LAST UPDTAED              BYTE

```

```

0115 1 ! SYDATE (9)      CURRENT DATE READ FROM SYSTEM      BYTE
0116 1 ! SYTIME (8)     CURRENT TIME READ FROM SYSTEM      BYTE
0117 1 ! TMP            TEMPERATURE                        REAL*4
0118 1 ! UMKZ          BOTTOM BACK SCATTERING COEF.        REAL*4      -28.0
0119 1 ! WS            WIND SPEED                          REAL*4
0120 1 ! Z             (50)    DEPTH OF POINT OF SOUND SPEED  REAL*4
0121 1 ! ZZ            (50)    DEPTH OF POINT OF SOUND SPEED  REAL*4
0122 1
0123 1      INTEGER*2 ITO,MGSOP,N,NN
0124 1      REAL*4     BDF,BIOP,C(50),CC(50),CS,DEG,EL,F,GRDS
0125 1      REAL*4     PI,TMP,UMKZ,WS,Z(50),ZZ(50)
0126 1      BYTE       SYDATE(9),SYTIME(8),BTDATE(9),BTTIME(8)
0127 1      BYTE       SNDATE(9),SNTIME(8)
0128 1      DATA      PI,DEG,GRDS/3.1415927,57.2957795,0.0164/
0129 1      DATA      UMKZ/-28./
0130 1
0131 1      COMMON /SVP/ F,N,Z,C,EL,MGSOP,BDF,WS,CS,TMP,BIOP,
0132 1      1          UMKZ,PI,DEG,GRDS,ITO,ZZ,CC,NN,
0133 1      2          SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME
0134 1 ! -----SVP-END-----
0135      INCLUDE 'SVP1.INC'
0136 1 ! -----SVP1-----
0137 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0138 1 ! -----
0139 1 ! BUFFER (224)  HISTORICAL DATA FILE BUFFER          REAL*4
0140 1 ! DS          (30)  HISTORICAL DEPTH                  REAL*4
0141 1 ! J20          # OF DEEP OCEAN DEPTH/VEL PAIRS        INTEGER*2
0142 1 ! NS          TOTAL # OF PAIRS IN HISTORICAL          INTEGER*2
0143 1 ! NSN         MONTH NUMBER (1=JAN.,ETC)                INTEGER*2  1 TO 12
0144 1 ! SLNTY       SALINITY                                REAL*4
0145 1 ! VS          (30)  HISTORICAL VELOCITY               REAL*4
0146 1
0147 1      REAL*4     BUFFER,DS,SLNTY,VS
0148 1      INTEGER*2  J20,NSN,NS
0149 1
0150 1      COMMON /SVP1/ J20,BUFFER(224),NSN,SLNTY,DS(30),VS(30),NS
0151 1 ! -----END SVP1-----
0152 1
0153 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0154 1 ! -----
0155 1 ! I      COUNTER                                INTEGER*2
0156 1 ! IDD    DAY IN DATE                            INTEGER*2
0157 1 ! IFIN   POINTER FINISH OF IROC ARRAY            INTEGER*2
0158 1 ! IPRINT PRINT FILE FLAG                        INTEGER*2  Y OR N
0159 1 ! IROC   (15)  ARRAY WITH NNAMES OF OCEANS          INTEGER*2
0160 1 ! INPBDF INPUTTED BOTTOM DEPTH (FATHOMS)            INTEGER*2
0161 1 ! INSSP   SSP ENTRY TYPE - MANUAL OR HIST          INTEGER*2
0162 1 ! ISTR    POINTER START IN IROC ARRAY              INTEGER*2
0163 1 ! IYY     YEAR IN DATE                            INTEGER*2
0164 1 ! J      COUNTER                                INTEGER*2
0165 1 ! JANS    OPERATOR RESPONSE                        INTEGER*2
0166 1 ! K      COUNTER                                INTEGER*2
0167 1 ! MAPFLG  FLAG FOR MAP FORM TASK                    INTEGER*2
0168 1 ! MONTH   MONTH OF THE YEAR                        INTEGER*2
0169 1 ! NBT     NUMBER OF BT POINTS                       INTEGER*2
0170 1 ! NEWBT   FLAG FOR INPUTTING OF NEW BT              INTEGER*2
0171 1 ! NHIST   FLAG FOR HISTORICAL SSP ENTRY            INTEGER*2

```

```

0172 ! NP          NUMBER OF BT POINTS          INTEGER*2
0173 ! NUM        NUMBER OF BT POINTS          INTEGER*2
0174 ! SUM        FACTOR FOR DEPTH CALCULATIONS  REAL*4
0175 ! ZB         DEPTH IN FEET                 REAL*4
0176 !
0177 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0178
0179          INTEGER*2 I,IDD,IFIN,INPBDF,INSSP,IPRINT,IROC(15),ISTR,IYY
0180          INTEGER*2 J,JANS,K,MAPFLG,MONTH,NBT,NEWBT,NHIST,NP,NUM
0181          REAL*4    SUM,ZB
0182          DATA     IROC    /'N.','PA','C.','NL','AN','T.','ME','DS',
0183          1          'EA','IN','DI','AN','NO','RS','EA'/
0184
0185 ! -----PRELIMINARIES-----
0186          MAPFLG=.TRUE.          ! INITIALIZE MAP FLAG
0187          CLOSE(UNIT=6)          ! CLOSE PRINT UNIT
0188          IF(IPRINT.EQ.'Y')OPEN(UNIT=6,NAME='ENVIRN.LST;1',DISP='PRINT',
0189 1          STATUS='UNKNOWN')      ! OPEN PRINT UNIT
0190          IF(IPRINT.EQ.'N')OPEN(UNIT=6,NAME='ENVIRN.LST;1',STATUS='UNKNOWN')
0191          CALL ICLR              ! CLEARS SCREEN
0192
0193          READ(2,1)N,(Z(I),C(I),I=1,N),DLYR,! READ LAST STORED SSP DATA
0194 1          MGS,BDF,WS,CS,TMP,BIO,UMKZ,LAT,LONG,NOC,INDX,RCZ,NSN,
0195 2          NMAREA,SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME,
0196 3          SLNTY,ILYR,NNBT,(DD(I),TT(I),I=1,NNBT),
0197 4          NN,(ZZ(I),CC(I),I=1,NN),INPBDF,ISVP,
0198 5          INBT,(DBT(I),VBT(I),I=1,INBT)
0199          EL=DLYR                ! CHANGE 'EL' TO 'DLYR'
0200          MGSOP=MGS              ! CHANGE 'MGSOP' TO 'MGS'
0201          ZB=6.*BDF              ! ASSIGN DEPTH IN FEET
0202          CALL ICLR              ! CLEAR SCREEN
0203          ISTR=(NOC-1)*3+1        ! START OF IROC ARRAY
0204          IFIN=ISTR+2            ! FINISH OF IROC ARRAY
0205
0206          WRITE(5,145)BTDATE(1),BTDATE(2),BTTIME(1),BTTIME(2),BTTIME(4),
0207 1          BTTIME(5),BTDATE(4),BTDATE(5),BTDATE(6),BTDATE(8),BTDATE(9),
0208 2          (IROC(I),I=ISTR,IFIN),INDX      ! DISPLAY LAST SSP DATA
0209
0210          READ(5,1050) JANS        ! NEW BT/SSP DATA QUERY
0211          IF(JANS.NE.'Y') GO TO 340 ! NEW BT/SSP RESPONSE
0212          CALL DATE(SYDATE)        ! GET SYSYEM DATE
0213          CALL TIME(SYTIME)        ! GET SYSTEM TIME
0214          CALL MAP(MAPFLG)        ! GET MODULES IN TASK
0215          CALL ICLR              ! CLEAR SCREEN
0216          WRITE(5,1475)           ! ASK FOR TYPE OF SSP ENTRY
0217          READ(5,1360) INSSP      ! MANUAL OR HIST SELECTION
0218
0219 ! -----MANUAL-----
0220          CALL ICLR              ! CLEAR SCREEN
0221          NBT=0                  ! INITIALIZE # OF XBT POINTS
0222          INBT=0                  ! INITIALIZE # OF XBT POINTS
0223          CALL KEYPCH(INSSP,NBT,MAPFLG) ! GET OPERATOR INPUT PAIRS
0224          ISVP=2                  ! SVP CHOICE 2
0225          INPBDF=INT(BDF+0.5)     ! INPUT BOTTOM DEPTH
0226          IF(INSSP.EQ.2) GO TO 320 ! INPUTTED SSP CHOICE IS 2
0227 282          CALL XBT(INSSP,NBT,NHIST,NEWBT) ! CORRECT POSSIBLE BT ERRORS
0228          IF(NEWBT.EQ.1) THEN      ! NEW BT NEEDED

```

```

0229      CALL BT(INSSP,NBT)      ! GET OPERATOR INPUT AGAIN
0230      GOTO 282                ! GO BACK TO CALL XBT
      B1      END IF              ! END IF BLOCK
0232      IF(NHIST.EQ.1) THEN      ! SKIP TO HISTORICAL PART
0233
0234      !-----HISTORICAL-----
0235      320      ISVP=1            ! SET ISVP TO ONE
0236      INBT=0                    ! SO SVFGRF WON'T DISPLAY LAST XBT
0237      DO 335 I=1,NS            ! DO FOR NUMBER OF BTS
0238      Z(I)=DS(I)                ! DEPTH
0239      C(I)=VS(I)                ! VELOCITY
0240      335      CONTINUE          ! END DO LOOP
0241      N=NS                      ! NUMBER OF DEPTH/VELOCITY PAIRS
0242      END IF                    ! END IF BLOCK
0243      !-----CORRECTED VALUES-----
0244      ZB = 6.*BDF               ! ESTIMATED BOTTOM DEPTH
0245      CALL INSERT(N,Z,C,ZB,NUM) ! INSERT DEPTH INTO SSP
0246      N=NUM                     ! SET NUMBER OF BTS
0247      IF(NUM.GT.50) THEN        ! > 50 BTS
0248      Z(50)=ZB                  ! DEPTH OF 50TH BT
0249      C(50)=C(48)+(ZB-Z(48))/(Z(49)-Z(48))*(C(49)-C(48)) ! VELOCITY
0250      N=50                      ! NUMBER OF BT
0251      END IF                    ! END IF BLOCK
0252      SUM=0.                     ! SET SUM TO ZERO
0253      CALL DUPDEP(N,Z,C)         ! CHECK FOR DUPLICATE DEPTHS
0254      CALL DUFVEL(N,Z,C)         ! CHECK FOR DUPLICATE VELOCITY
0255      DO 350 I=2,N              ! DO FOR NUMBER OF BT
0256      SUM=SUM+(Z(I)-Z(I-1))*(C(I)+C(I-1)) ! SUM IN FT*FT/SEC
0257      350      CONTINUE          ! END DO LOOP
0258      SUM=0.5*SUM/Z(N)           ! EVALUATED SUM IN FT/SEC
0259      BDF=BDF*SUM/4800.          ! CORRECTED BOTTOM (FATHOMS)
0260      ZB=6.*BDF                 ! CORRECTED BOTTOM DEPTH(FT)
0261
0262      C(N)=C(N-1)+(C(N)-C(N-1))*(ZB-Z(N-1))/(Z(N)-Z(N-1)) ! VELOCITY
0263      Z(N)=ZB                    ! CORRECTED DEPTH
0264      CS=C(1)                    ! SURFACE VELOCITY
0265      CALL VELTMP(Z(1),C(1),TMP,SLNTY) ! GET SURFACE TEMPERATURE
0266      IF(TMP.LE.1.) TMP=0.       ! NO NEGATIVE TEMPS ALLOWED
0267      CALL LAYER(N,Z,C,DLYR)     ! LOCATE LAYER DEPTH
0268      EL=DLYR                    ! LAYER DEPTH
0269      CALL NOCONV(RCZ,ILYR)       ! FIND CZ RANGE
0270      CALL DUPDEP(N,Z,C)         ! CHECK FOR DUPLICATE DEPTHS
0271      CALL DUFVEL(N,Z,C)         ! CHECK FOR DUPLICATE VELOCITY
0272      340      WRITE(6,2800)      ! 'NEW DATA TO BE STORED'
0273      CALL ICLR                  ! CLEAR SCREEN
0274      CALL SVFGRF(INPBDF)        ! GRAPHIC DISPLAY OF SVP
0275      CALL ICLR                  ! CLEAR SCREEN
0276
0277      WRITE(2'1)N,(Z(I),C(I),I=1,N),EL,MGSOP,BDF,WS,
0278      1      CS,TMP,BIO,UMKZ,LAT,LONG,NOC,INDX,RCZ,NSN,NMAREA,
0279      2      SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME,
0280      3      SLNTY,ILYR,NNBT,(DD(I),TT(I),I=1,NNBT),NN,
0281      4      (ZZ(I),CC(I),I=1,NN),INPBDF,ISVP, ! WRITE NEW DATA
0282      5      INBT,(DBT(I),VBT(I),I=1,INBT)
0283
0284      CLOSE(UNIT=6)              ! CLOSE PRINT UNIT 6
      )5      RETURN              ! RETURN TO CALLING ROUTINE

```



```

0286
0287 !-----FORMAT STATEMENTS-----
0288 145  FORMAT(5X,' LAST BT TAKEN ON',2X,6A1,'Z ',3A1,' ',2A1,
0289      1 ' IN ',3A2,' AREA:',I2
0290      2 /5X,' IF BT DATA IS MORE THAN 4 HOURS OLD, NEW BT NEEDED' /
0291      3 /5X,' DO YOU WISH TO ENTER NEW BT/SSP DATA?'
0292      4 /4X,' ****ANSWER YES OR NO****',T60,' ')
0293      800  FORMAT(/4X,' **** ENTER TRUE WIND SPEED (XX KTS) ****',T60,' ')
0294      850  FORMAT(F10.0)
0295      1050  FORMAT(A1)
0296      1360  FORMAT(I4)
0297      1475  FORMAT(5X,' ****SELECT TYPE OF SSP DESIRED****'
0298      1      ///8X,' 1 = MANUAL BT'
0299      2      /8X,' 2 = HISTORICAL SSP FOR AREA AND SEASON',T60,' ')
0300      2800  FORMAT(1H1,T22,' NEW DATA TO BE STORED:')
0301      END

```

## COMMAND QUALIFIERS

```
FORTRAN /CHECK=ALL/LIST/SHOW=(INCLUDE,NOMAP) [LAFLEUR]ENVIRN.F77
```

```
/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
```

```
/DEBUG=(NOSYMBOLS,TRACEBACK)
```

```
/STANDARD=(NOSYNTAX,NOSOURCE_FORM)
```

```
/SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)
```

```
/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD_LINES /NOCROSS_REFERENCE /I
```

## COMPILATION STATISTICS

```
Run Time: 5.17 seconds
```

```
Elapsed Time: 7.19 seconds
```

```
Page Faults: 467
```

```
Dynamic Memory: 185 pages
```

```

0001             INTEGER*2 FUNCTION FLOOR-REAL)
0002
0003 ! PROLOGUE:
0004 ! MODULE NAME: FLOOR
0005 ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006 ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007 ! DATE: 1982 & 6/84 (FORTRAN 77)
0008 ! FUNCTION: FUNCTION FLOOR DETERMINES THE MAXIMUM INTEGER*2
0009 !             THAT IS LESS THAN OR EQUAL TO THE REAL VALUE PASSED
0010 !             TO THE FUNCTION.
0011 ! INPUTS: REAL NUMBER PASSED TO FIND FLOOR OF
0012 ! OUTPUTS: MAXIMUM INTEGER*2 <= REAL NUMBER PASSED IN
0013 ! MODULES CALLED: NONE
0014 ! CALLED BY: MAP
0015 !
0016 ! VARBL  SIZE      PURPOSE                                TYPE      RANGE
0017 ! -----
0018 ! I              TRUNCATED REAL VALUE                    INTEGER*2
0019 ! REAL          A REAL VALUE                              REAL*4
0020 !
0021             INTEGER*2 I
0022             REAL*4 REAL
0023
0024             I=INT-REAL)                                ! TRUNCATE REAL VALUE
0025             FLOOR=I+(REAL.LT.0.)                        ! IF REAL NEG, OFFSET
0026             IF (FLOATI(I).EQ.REAL) FLOOR=I              ! NUMBERS EQUAL, NO OFFSET
0027
0028             RETURN                                        ! RETURN TO CALLING ROUTINE
0029             END                                          ! END SUBROUTINE

```

```

0001      INTEGER*2 FUNCTION FNP(I)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: FNP
0005      ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006      ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007      ! DATE: 1982 & 6/84 (FORTRAN 77)
0008      ! FUNCTION: FUNCTION FIND POINT IS DESIGNED TO GET THE
0009      !              LATITUDE/LONGITUDE VALUE OF POINT I.
0010      ! INPUTS: PARAMETER I PASSED IN
0011      ! OUTPUTS: LATITUDE/LONGITUDE VALUE OF POINT I.
0012      ! MODULES CALLED: NONE
0013      ! CALLED BY: DNUT, GRAPH
0014      !
0015      INCLUDE 'MAP.PAR'
0016      1      PARAMETER STOLEN=3800
0017      1      PARAMETER SEGLEN=60, POLLEN=40
0018      1      PARAMETER WRKLEN=1000, NDXLEN=300
0019      1      PARAMETER MAXDTY=3
0020      1      PARAMETER TOL=3
0021      1      PARAMETER DEG=57.2957795
0022      1      PARAMETER RAD=.017453293
0023      1      PARAMETER PI=3.14159265
0024      1      PARAMETER ERAD=3440.3
0025      1      PARAMETER S251=63001
0026      1      PARAMETER TWO15=32768
0027      1
0028      1 !      INTEGER*2 MAXDTY,NDXLEN,POLLEN,SEGLEN,STOLEN,TOL,WRKLEN
0029      1 !      INTEGER*4 S251,TWO15
0030      1 !      REAL*4      DEG,ERAD,PI,RAD
0031      INCLUDE 'CS.INC'
0032      1 ! -----CS-----
0033      1 ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0034      1 ! -----
0035      1 ! S      -1,3800    POLYGON AND SEGMENT STORAGE ARRAY      REAL*4
0036      1 ! STOLEN          STORAGE ARRAY LENGTH (FOR SEGS & POLYS) PARM
0037      1 !
0038      1      REAL*4  S(-1:STOLEN)
0039      1
0040      1      COMMON /CS/      S
0041      1 ! -----CS-END-----
0042      1
0043      !
0044      ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0045      ! -----
0046      ! I          INDEX FOR THE POINT VALUE                    INTEGER*2
0047      ! J          ABSOLUTE VALUE OF I                          INTEGER*2
0048      !
0049      ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0050
0051      INTEGER*2 I,J
0052
0053      FNP=0      ! INITIALIZE FIND POINT
0054      IF (I.GT.12) FNP=I+4      ! INDEX OF 1ST END PT OF SEGMENT
0055      J=ABS(I)      ! MUST BE POSITIVE VALUE
0056      IF (I.LT.-12) FNP=I+FIX(S(J))-I+3 ! SET TO LAST PT OF SEGMENT
0057      IF (FNP.NE.0) GO TO 999      ! RETURN TO CALLING PROGRAM
0058      WRITE(5,102)      ! ERROR MESSAGE
0059      999      RETURN      ! RETURN TO CALLING ROUTINE

```

```

0060
0061 !-----FORMAT STATEMENT-----
0062 102  FORMAT(X,'PROGRAM ERROR, ATTEMPT TO USE FNP ON BORDER')
0063      END

```

#### COMMAND QUALIFIERS

```

FORTRAN /CHECK=ALL/LIST/SHOW=(INCLUDE,NOMAP) [LAFLEUR]FNP.F77

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)
/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD_LINES /NOCROSS_REFERENCE

```

#### COMPILATION STATISTICS

```

Run Time:          0.87 seconds
Elapsed Time:      1.73 seconds
Page Faults:       364
Dynamic Memory:    123 pages

```

```

0001      SUBROUTINE FORCST(IPRINT)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: FORCST
0005      ! AUTHOR: G. BROWN & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1974 & 11/83 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE FORCST OBTAINS ENVIRONMENTAL PARAMETERS
0008      !               FROM HISTORICAL DATA FOR THE AREA AND MONTH.
0009      ! INPUTS: INPUTTED PARAMETERS. VARIABLES IN COMMONS.
0010      ! OUTPUTS: FORECASTING DATA FILES.
0011      ! MODULES CALLED: DUPDEP,DUPVEL,ICLR,INSERT,KEYPCH,LAYER,MAP,
0012      !               NOCONV,SVPGRF,VELTMP,XBT
0013      ! CALLED BY: SIMAS
0014      !
0015      ! =====
0016      ! ALGORITHMS USED:
0017      !
0018      !   CORRECTION FACTOR=.5*SUMMATION FROM 2 TO N OF (Z(I)-Z(I-1))*C(I-1)
0019      !               DIVIDED BY Z(N) * 4800FT/SEC
0020      !
0021      !   TRUE DEPTH = (ESTIMATED BOTTOM DEPTH,Z(N)) * CORRECTION FACTOR
0022      !
0023      !               (C(N)-C(N-1))*(ZB-Z(N-1))
0024      !   SOUND VELOCITY FOR TRUE DEPTH = C(N-1) + -----
0025      !               Z(N) - Z(N-1)
0026      !   WHERE ZB IS THE TRUE DEPTH
0027      ! =====
0028      !
0029      ! INCLUDE 'DHST.INC'
0030      ! -----DHST-----
0031      ! VARBL  SIZE  PURPOSE  TYPE  RANGE
0032      ! -----
0033      ! SCHNLD          SOUND CHANNEL LAYER DEPTH  REAL*4
0034      !
0035      ! REAL*4  SCHNLD
0036      !
0037      ! COMMON /DHST/ SCHNLD
0038      ! -----DHST END-----
0039      ! INCLUDE 'DTV.INC'
0040      ! -----DTV-----
0041      ! VARBL  SIZE  PURPOSE  TYPE  RANGE
0042      ! -----
0043      ! D      (25)  DEPTH      REAL*4
0044      ! DD     (25)  DEPTH      REAL*4
0045      ! NNBT          NUMBER OF BATHETHERMAL  INTEGER*2
0046      ! T      (25)  TEMPERATURE  REAL*4
0047      ! TT     (25)  TEMPERATURE  REAL*4
0048      ! VEL    (25)  VELOCITY    REAL*4
0049      !
0050      ! INTEGER*2 NNBT
0051      ! REAL*4 D,DD,T,TT,VEL
0052      !
0053      ! COMMON /DTV/  D(25),T(25),VEL(25),DD(25),TT(25),NNBT
0054      ! -----END DTV-----
0055      ! INCLUDE 'ENVN.INC'
0056      ! -----ENVN-----
0057      ! VARBL  SIZE  PURPOSE  TYPE  RANGE

```

FORCST

14-Dec-1984 08:32:17

14-Dec-1984 08:32:11

```

0058 1 ! -----
0059 1 ! BIO      (2)      BIOLOGICAL BACK SCATTERING REAL*4      -57. & -47.
0060 1 ! DLVR      LAYER DEPTH      REAL*4
0061 1 ! MGS      MGS PROVINCE      INTEGER*2
0062 1
0063 1      REAL*4      BIO,DLVR
0064 1      INTEGER*2 MGS
0065 1      DATA BIO/-57.,-47./
0066 1
0067 1      COMMON /ENVN/ BIO(2),DLVR,MGS
0068 1
0069 1 ! -----END ENVN-----
0070      INCLUDE 'GRF.INC'
0071 1 ! -----GRF-----
0072 1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0073 1 ! -----
0074 1 ! DBT      (25)  DEPTH OF DEPTH/VEL PAIR      REAL*4
0075 1 ! IANS      PREDICTION TYPE      INTEGER*2      -2 TO +2
0076 1 ! ILVR      INDEX FOR LAYER DEPTH      INTEGER*2
0077 1 ! INBT      OPERATOR ENTERED # OF BT POINTS      INTEGER*2
0078 1 ! ISVP      LATEST OR HISTORICAL BT FLAG      INTEGER*2      1 OR 2
0079 1 ! I2000     SVP INDEX FOR 2000 FT DEPTH      INTEGER*2
0080 1 ! VBT      (25)  VELOCITY FOR DEPTH PAIR REAL*4      REAL*4
0081 1
0082 1      REAL*4      DBT,VBT
0083 1      INTEGER*2 IANS,ILVR,INBT,ISVP,I2000
0084 1
0085 1      COMMON /GRF/ IANS,ISVP,ILVR,I2000,INBT,DBT(25),VBT(25)
0086 1
0087 1 ! -----END GRF-----
0088      INCLUDE 'LOC.INC'
0089 1 ! -----LOC-----
0090 1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0091 1 ! -----
0092 1 ! INDX      SSP INDEX      INTEGER*2
0093 1 ! LAT      (4)  LATITUDE      INTEGER*2
0094 1 ! LONG     (4)  LONGITUDE      INTEGER*2
0095 1 ! NMAREA   (20)  AREA OCEAN NAME      BYTE
0096 1 ! NOC      NUMBER OF OCEAN      INTEGER*2
0097 1 ! RCZ      RANGE TO CONVERG. ZONE REAL*4
0098 1
0099 1      REAL*4      RCZ
0100 1      INTEGER*2 INDX,LAT,LONG,NOC
0101 1      BYTE      NMAREA(20)
0102 1
0103 1      COMMON /LOC/ LAT(4),LONG(4),NOC,INDX,RCZ,NMAREA
0104 1
0105 1 ! -----END LOC-----
0106      INCLUDE 'SVP.INC'
0107 1 ! -----SVP-----
0108 1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0109 1 ! -----
0110 1 ! BDF      BOTTOM DEPTH IN FATHOMS      REAL*4
0111 1 ! BIOP     BIOLOGICAL BACK SCATTERING COEF REAL*4
0112 1 ! BTDATE   (9)  DATE OF LAST BT INPUT      BYTE
0113 1 ! BTTIME   (8)  TIME OF LAST BT INPUT      BYTE
0114 1 ! C        (50)  VELOCITY (PAIRED WITH Z FOR SVP) REAL*4

```

C-20.2

```

0115 1 ! CC      (50)  VELOCITY (PAIRED WITH ZZ FOR SVP) REAL*4
0116 1 ! CS              SOUND VELOCITY AT SURFACE      REAL*4
0117 1 ! DEG            TEMPERATURE (DEG)                REAL*4      57.2957795
0118 1 ! EL              LAYER DEPTH                     DATA
0119 1 ! F                FREQUENCY                     REAL*4
0120 1 ! GRDS            GRIDS                           REAL*4      0.0164
0121 1 ! ITO             MINIMAL 2-WAY TRAVEL TIME        INTEGER*2
0122 1 ! MGSOP           MGS PROVINCE NUMBER             INTEGER*2
0123 1 ! N                # OF DEPTH/VELOCITY PAIRS       INTEGER*2
0124 1 ! NN              # OF DEPTH/VELOCITY PAIRS       INTEGER*2
0125 1 ! PI              MATHEMATICAL CONSTANT PI        REAL*4      3.1415927
0126 1 ! SNDATE (9)      DATE SYS PARMS LAST UPDATED     BYTE
0127 1 ! SNTIME (8)      TIME SYS PARMS LAST UPDTAED     BYTE
0128 1 ! SYDATE (9)      CURRENT DATE READ FROM SYSTEM   BYTE
0129 1 ! SYTIME (8)      CURRENT TIME READ FROM SYSTEM   BYTE
0130 1 ! TMP             TEMPERATURE                     REAL*4
0131 1 ! UMKZ            BOTTOM BACK SCATTERING COEF.     REAL*4      -28.0
0132 1 ! WS              WIND SPEED                     REAL*4
0133 1 ! Z                (50)  DEPTH OF POINT OF SOUND SPEED REAL*4
0134 1 ! ZZ              (50)  DEPTH OF POINT OF SOUND SPEED REAL*4
0135 1
0136 1      INTEGER*2 ITO,MGSOP,N,NN
0137 1      REAL*4    BDF,BIOP,C(50),CC(50),CS,DEG,EL,F,GRDS
0138 1      REAL*4    PI,TMP,UMKZ,WS,Z(50),ZZ(50)
0139 1      BYTE      SYDATE(9),SYTIME(8),BTDATE(9),BTTIME(8)
0140 1      BYTE      SNDATE(9),SNTIME(8)
0141 1      DATA     PI,DEG,GRDS/3.1415927,57.2957795,0.0164/
0142 1      DATA     UMKZ/-28./
0143 1
0144 1      COMMON /SVP/ F,N,Z,C,EL,MGSOP,BDF,WS,CS,TMP,BIOP,
0145 1      1          UMKZ,PI,DEG,GRDS,ITO,ZZ,CC,NN,
0146 1      2          SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME
0147 1 ! -----SVP-END-----
0148 1      INCLUDE 'SVP1.INC'
0149 1 ! -----SVP1-----
0150 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0151 1 ! -----
0152 1 ! BUFFER (224)  HISTORICAL DATA FILE BUFFER          REAL*4
0153 1 ! DS      (30)  HISTORICAL DEPTH                      REAL*4
0154 1 ! J20              # OF DEEP OCEAN DEPTH/VEL PAIRS    INTEGER*2
0155 1 ! NS              TOTAL # OF PAIRS IN HISTORICAL      INTEGER*2
0156 1 ! NSN            MONTH NUMBER (1=JAN.,ETC)            INTEGER*2  1 TO 12
0157 1 ! SLNTY          SALINITY                             REAL*4
0158 1 ! VS      (30)  HISTORICAL VELOCITY                  REAL*4
0159 1
0160 1      REAL*4    BUFFER,DS,SLNTY,VS
0161 1      INTEGER*2 J20,NSN,NS
0162 1
0163 1      COMMON /SVP1/ J20,BUFFER(224),NSN,SLNTY,DS(30),VS(30),NS
0164 1 ! -----END SVP1-----
0165 1
0166 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0167 1 ! -----
0168 1 ! I                COUNTER                              INTEGER*2
0169 1 ! INPBDF          INPUTTED BOTTOM DEPTH                 INTEGER*2
0170 1 ! INSSP          DATA INPUT SELECTION                 INTEGER*2
0171 1 ! IPRINT         PRINT FLAG                            INTEGER*2

```

```

0172      ! J          COUNTER          INTEGER*2
0173      ! K          COUNTER          INTEGER*2
0174      ! MAPFLG     FLAG FOR MAP TASK  INTEGER*2
0175      ! NBT        NUMBER OF BATHETHERMAL  INTEGER*2
0176      ! NEWBT      FLAG FOR NEW BT DATA  INTEGER*2
0177      ! NHIST      FLAG FOR HISTORICAL BT DATA  INTEGER*2
0178      ! NP         NUMBER OF BT + 1      INTEGER*2
0179      ! NUM        # OF POINTS TO INSERT IN SVP  INTEGER*2
0180      ! SUM        FACTOR IN DEPTH EQUATION  INTEGER*2
0181      ! ZB         BOTTOM DEPTH IN FEET      REAL*4
0182      !
0183      ! *** VARIABLES NOT LISTED HERE ARE IN COMMON ENVN,GRF,LOC,SVP,SVPl **
0184
0185      INTEGER*2 MAPFLG
0186      INTEGER*2 I,INPBDF,INSSP,IPRINT,J,K
0187      INTEGER*2 NBT,NEWBT,NHIST,NP,NUM
0188      REAL*4     SUM,ZB
0189
0190      !-----PRELIMINARIES-----
0191      MAPFLG=.TRUE.          ! INITIALIZE MAP FLAG
0192      CALL ICLR              ! CLEAR SCREEN
0193      CLOSE(UNIT=6)         ! CLOSE UNIT 6
0194      IF(IPRINT.EQ.'Y')OPEN(UNIT=6,NAME='FORCST.LST;1',DISP='PRINT',
0195      1      STATUS='UNKNOWN') ! PRINT OPTION
0196      IF(IPRINT.EQ.'N')OPEN(UNIT=6,NAME='FORCST.LST;1',STATUS='UNKNOWN')
0197      CALL ICLR              ! CLEAR SCREEN
0198                          ! READ FORECASTING DATA FILE
0199      READ(2;1) N,(Z(I),C(I),I=1,N),DLYR,MGS,BDF,WS,
0200      1      CS,TMP,BIO,UMKZ,LAT,LONG,NOC,INDX,RCZ,NSN,NMAREA,
0201      2      SYDATE,SYTIME,BTDATE,BTIME,SNDATE,SNTIME,
0202      3      SLNTY,ILYR,NNBT,(DD(I),TT(I),I=1,NNBT),
0203      4      NN,(ZZ(I),CC(I),I=1,NN),INPBDF,ISVP,
0204      5      INBT,(DBT(I),VBT(I),I=1,INBT)
0205
0206      EL=DLYR                ! CHANGE EL TO DLYR THRU SIMAS
0207      MGSOP=MGS              ! CHANGE MGSOP TO MGS THRU SIMAS
0208      100 WRITE(5,1400)      ! SELECT DATA INPUT DESIRED
0209      READ(5,1360) INSSP     ! READ SELECTION
0210      IF(INSSP.EQ.1) GOTO 9999 ! USE EXISTING DATA FILE
0211      IF(INSSP.LT.1.OR.INSSP.GT.5) GOTO 100 ! INVALID, LOOP BACK
0212
0213      !-----SELECT SVP & SET MISC. PARMS-----
0214      IF(INSSP.EQ.2)         ! AUTOMATED HISTORY DATA ENTRY
0215      1      CALL MAP(MAPFLG) ! INITIATE THE MAP TASK
0216      NBT=0                 ! INITIALIZE # OF BT
0217      INBT=0                ! INITIALIZE INPUTTED # OF BT
0218      CALL KEYPCH(INSSP,NBT,MAPFLG) ! OPERATOR INPUT SPECIFIC PROFILE
0219      IF(INSSP.EQ.4) ISVP=2   ! SET FLAG FOR LATEST XBT LABEL
0220      IF(INSSP.EQ.5) ISVP=5   ! KEYPUNCH OPTION
0221      INPBDF=INT(BDF+0.5)     ! INPUTTED BOTTOM DEPTH
0222      IF(INSSP.EQ.5) GOTO 590 ! KEYPUNCH ENTIRE SSP & DATA CHOICE
0223      CALL ICLR              ! CLEAR SCREEN
0224      IF(INSSP.EQ.2.OR.INSSP.EQ.3) GOTO 333 ! CHOICE 2 AND 3
0225      282 CALL XBT(INSSP,NBT,NHIST,NEWBT) ! XBT ERROR CORRECTING
0226      IF(NEWBT.EQ.1) THEN    ! IF TRUE, ENTER NEW BT
0227      CALL BT(INSSP,NBT)     ! ENTER NEW XBT
0228      8      GOTO 282        ! CHECK NEW XBT

```



```

0229      END IF                                ! END IF BLOCK
0230
0231      !-----HISTORICAL SSP-----
0232      IF(NHIST.EQ.1) THEN                      ! IF TRUE, USE HIST ONLY
0233      333      ISVP=1                          ! SET FLAG FOR HISTORICAL LABEL
0234              INBT=0                          ! SO SVPGRF WON'T DISPLAY LAST XBT
0235              DO 335 I = 1,NS                  ! DO FOR NUMBER OF BT
0236                  Z(I) = DS(I)                ! HISTORICAL DEPTH
0237                  C(I) = VS(I)                ! HISTORICAL VELOCITY
0238      335      CONTINUE                        ! END DO LOOP
0239              N=NS                            ! NUMBER OF DEPTH/VELOCITY PAIRS
0240      END IF                                ! END IF BLOCK
0241
0242      !-----INSERT ESTIMATED BOTTOM DEPTH-----
0243      ZB=6.*BDF                                ! BOTTOM DEPTH IN FEET
0244      CALL INSERT(N,Z,C,ZB,NUM)                ! INSERT DEPTH/VEL POINT INTO SVP
0245      N=NUM                                    ! SET N
0246      IF(NUM.GT.50) THEN                      ! IF > 50 BT POINTS
0247          Z(50)=ZB                            ! DEPTH
0248          C(50)=C(48)+(ZB-Z(48))/(Z(49)-Z(48))*(C(49)-C(48)) ! DEPTH
0249          NUM=50                              ! MAX NUMBER IS 50
0250      END IF                                ! END IF BLOCK
0251      SUM=0.                                  ! INITIALIZE SUM
0252      CALL DUPDEP(N,Z,C)                      ! CHECK FOR DUPLICATE DEPTHS
0253      CALL DUPVEL(N,Z,C)                      ! CHECK FOR DUPLICATE VELOCITY
0254      DO 350 I=2,N                            ! CORRECTIONS FOR TRUE DEPTH
0255          SUM=SUM+(Z(I)-Z(I-1))*(C(I)+C(I-1)) ! SUM IN FT**2/SEC
0256      350      CONTINUE                        ! END DO LOOP
0257      SUM=0.5*SUM/Z(N)                        ! SUM IN FT/SEC
0258      BDF=BDF*SUM/4800.                      ! CORRECTED DEPTH IN FATHOMS
0259      ZB=6.*BDF                              ! CORRECTED DEPTH IN FEET
0260
0261      !-----CORRECTED VEL & DEPTH ENTERED IN SSP
0262      C(N)=C(N-1)+(C(N)-C(N-1))*(ZB-Z(N-1))/(Z(N)-Z(N-1)) ! VELOCITY
0263      Z(N)=ZB                                ! DEPTH
0264      590      CONTINUE                        ! NEEDED FOR GOTO STATEMENT
0265      CALL VELTMP(Z(1),C(1),TMP,SLNTY) ! OBTAIN TEMP AT SURFACE
0266      IF(TMP.LE.1.)TMP=0.                    ! MINIMUM TEMPERATURE
0267      CS=C(1)                                ! SOUND VELOCITY AT SURFACE
0268      CALL LAYER(N,Z,C,DLYR)                  ! LOCATE LAYER DEPTH IN SVP
0269      EL=DLYR                                ! CHANGE EL TO DLYR IN SIMAS
0270      CALL NOCONV(RCZ,ILYR)                    ! FIND RANGE TO CONVERG ZONE
0271      CALL DUPDEP(N,Z,C)                      ! CHECK FOR DUPLICATE DEPTHS
0272      CALL DUPVEL(N,Z,C)                      ! CHECK FOR DUPLICATE VELOCITY
0273      9999      CALL ICLR                      ! CLEAR SCREEN
0274              CALL SVPGRF(INPBDF)            ! PRODUCE GRAPH OF SVP
0275              CALL ICLR                      ! CLEAR SCREEN
0276              WRITE(2,1)                     ! WRITE FORECASTING DATA FILE
0277      1      N,(Z(I),C(I),I=1,N),DLYR,MGS,BDF,WS,
0278      1      CS,TMP,BIO,UMKZ,LAT,LONG,NOC,INDX,RCZ,NSN,NMAREA,
0279      2      SYDATE,SYTIME,BTDATE,BTIME,SNDATE,SNTIME,
0280      3      SLNTY,ILYR,NNBT,(DD(I),TT(I),I=1,NNBT),
0281      4      NN,(ZZ(I),CC(I),I=1,NN),INPBDF,ISVP,
0282      5      INBT,(DBT(I),VBT(I),I=1,INBT)
0283      CLOSE(UNIT=6)                          ! CLOSE FILE 6
0284      RETURN                                ! RETURN TO CALLING ROUTINE
0285
0286      )5

```

FORCST

14-Dec-1984 08:32:12  
14-Dec-1984 08:32:11

```
0286  !-----FORMAT STATEMENTS-----
0287  1360  FORMAT(I2)
0288  1400  FORMAT(' SELECT INPUT DESIRED:')
0289      1  ///,X,'1 = USE EXISTING FILE DATA',
0290      2  //,X,'2 = USE AUTOMATED HISTORICAL DATA ENTRY SELECTION',
0291      3  //,X,'3 = USE MANUAL HISTORICAL DATA ENTRY SELECTION',
0292      4  //,X,'4 = USE MANUAL BT/SSP ENTRY MERGED W/ HISTORICAL',
0293      5  //,X,'5 = KEYPUNCH ENTIRE SSP AND DATA',
0294      6  ///1H$,' **** ENTER YOUR CHOICE ****',T60,' ')
0295      END
```

#### COMMAND QUALIFIERS

FORTRAN /CHECK=ALL/LIST/SHOW=(INCLUDE,NOMAP) [LAFLEUR]FORCST.F77

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE\_FORM)

/SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)

/F77 /NOG\_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD\_LINES /NOCROSS\_REFERENCE /I

#### COMPILATION STATISTICS

Run Time: 4.68 seconds

Elapsed Time: 6.20 seconds

Page Faults: 452

Dynamic Memory: 180 pages

```

0001          SUBROUTINE FSETUP(RECNDX,R)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: FSETUP
0005      ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006      ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007      ! DATE: 1982 & 6/84 (FORTRAN 77)
0008      ! FUNCTION: THIS SUBROUTINE IS DESIGNED TO READ INFORMATION FROM THE
0009      !              FILE "MAP"(OAC)"A", CHECK THE LATITUDE & LONGITUDE RANGES,
0010      !              AND READ IN THE PROPER AMOUNT OF POINTER FOR THE FILE
0011      !              "MAP"(OAC)"B".
0012      ! INPUTS: DATA FROM FILES READ IN
0013      ! OUTPUTS: MESSAGES TO OPERATOR
0014      ! MODULES CALLED: OPNFIL
0015      ! CALLED BY: MAP
0016      !
0017          INCLUDE 'MAP.PAR'
0018      1      PARAMETER STOLEN=3800
0019      1      PARAMETER SEGLEN=60, POLLEN=40
0020      1      PARAMETER WRKLEN=1000, NDXLEN=300
0021      1      PARAMETER MAXDTY=3
0022      1      PARAMETER TOL=3
0023      1      PARAMETER DEG=57.2957795
0024      1      PARAMETER RAD=.017453293
0025      1      PARAMETER PI=3.14159265
0026      1      PARAMETER ERAD=3440.3
0027      1      PARAMETER S251=63001
0028      1      PARAMETER TWO15=32768
0029      1
0030      1 !      INTEGER*2 MAXDTY,NDXLEN,POLLEN,SEGLEN,STOLEN,TOL,WRKLEN
0031      1 !      INTEGER*4 S251,TWO15
0032      1 !      REAL*4      DEG,ERAD,PI,RAD
0033          INCLUDE 'CFILE.INC'
0034      1 ! -----CFILE.INC-----
0035      1 !  VARBL   SIZE      PURPOSE                                TYPE      RANGE
0036      1 !  -----
0037      1 !  FNAME   (21)      MAP FILE NAME                          CHAR
0038      1 !  OPEN    OPEN      OPEN FLAG                            LOGICAL*1 .FALSE.
0039      1 !
0040      1      LOGICAL*1 OPEN
0041      1      CHARACTER*1 FNAME(21)
0042      1
0043      1      COMMON /CFILE/ OPEN,FNAME
0044      1 ! -----END CFILE.INC-----
0045      1
0046          INCLUDE 'CL.INC'
0047      1 ! -----CL.INC-----
0048      1 !  VARBL   SIZE      PURPOSE                                TYPE      RANGE
0049      1 !  -----
0050      1 !  LATMAX          MAXIMUM LATITUDE                          INTEGER*2
0051      1 !  LATMIN          MINIMUM LATITUDE                          INTEGER*2
0052      1 !  LNGMAX          MAXIMUM LONGITUDE                          INTEGER*2
0053      1 !  LNGMIN          MINIMUM LONGITUDE                          INTEGER*2
0054      1 !
0055      1      INTEGER*2 LATMIN,LATMAX,LNGMIN,LNGMAX
0056      1
0057      1      COMMON /CL/ LATMIN,LATMAX,LNGMIN,LNGMAX
0058      1 ! -----END CL.INC-----
0059      1

```

```

0060 !
0061 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0062 ! -----
0063 ! A          .TRUE.                                BYTE  TRUE
0064 ! B          .FALSE.                               BYTE  FALSE
0065 ! I          COUNTER                                INTEGER*2
0066 ! J          NUMBER OF INDEXES                       INTEGER*2
0067 ! K          COUNTER                                INTEGER*2
0068 ! L          (4)  LAT;LNG MINIMUM;MAXIMUM ARRAY      INTEGER*2
0069 ! R          ERROR WITH FILE "MAP"(OAC)"A"          BYTE
0070 ! REC        NUMBER OF RECORDS COUNTER              INTEGER*2
0071 ! RECBUF (64) RECORD BUFFER                          INTEGER*2
0072 ! RECNDX (300) POINTERS INTO "MAP"(OAC)"B"          INTEGER*2
0073 ! RERR       NUMBER OF ERRORS IN INDEX FILE          INTEGER*2
0074 ! RTEST      TEST LAT;LNG VALIDITY                  BYTE
0075 ! W          NUMBER OF ITEMS READ FROM FILE          INTEGER*2
0076 !
0077 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0078
0079         INTEGER*2 I,J,K,REC,RECMAX,RERR,W
0080         INTEGER*2 RECNDX(NDXLEN),L(4),RECBUF(64)
0081         BYTE      RTEST,R,A,B
0082         EQUIVALENCE (LATMIN,L(1))
0083         DATA      A,B/.TRUE.,.FALSE./
0084
0085         R=.FALSE.                                ! DATA BASE ERROR FLAG
0086         CALL OPNFIL(A,R)                          ! OPEN MAP INDEX FILE
0087         IF (R) GO TO 999                          ! ERROR IN OPENING, RETURN
0088         READ (4'1,ERR=150) (L(I),I=1,4) ! READ MIN/MAX  LAT/LNG
0089
0090 !-----TEST DATA BASE DEFINED LATS, LNGS
0091         RTEST=(LATMIN.LT.-80 .OR. LATMIN.GT.80)! TEST MINIMUM LATITUDE
0092         R=RTEST+R                                ! SET ERROR FLAG
0093         IF (RTEST) WRITE(5,101)                  ! OUT OF RANGE ERROR
0094
0095         RTEST=(LATMAX.LT.-80 .OR. LATMAX.GT.80)! TEST MAXIMUM LATITUDE
0096         R=R+RTEST                                ! SET ERROR FLAG
0097         IF (RTEST) WRITE(5,102)                  ! OUT OF RANGE ERROR
0098
0099         RTEST=(LATMAX.LT.LATMIN)                  ! TEST LAT MAX > MIN
0100         R=R+RTEST                                ! SET ERROR FLAG
0101         IF (RTEST) WRITE(5,103)                  ! BOUNDARIES ERROR
0102
0103         RTEST=(LATMAX-LATMIN.LT.10)               ! TEST RANGE > 10
0104         R=R+RTEST                                ! SET ERROR FLAG
0105         IF (RTEST) WRITE(5,104)                  ! RANGE TOO NARROW
0106
0107         RTEST=(LATMAX-LATMIN.GT.100)              ! TEST RANGE < 100
0108         R=R+RTEST                                ! SET ERROR FLAG
0109         IF (RTEST) WRITE(5,105)                  ! RANGE TOO WIDE
0110
0111         RTEST=(LNGMIN.LT.-180 .OR. LNGMIN.GT.180) ! TEST MINIMUM LNG
0112         R=R+RTEST                                ! SET ERROR FLAG
0113         IF (RTEST) WRITE(5,106)                  ! OUT OF RANGE ERROR
0114
0115         RTEST=(LNGMAX.LT.-180 .OR. LNGMAX.GT.180) ! TEST MAXIMUM LNG
0116         R=R+RTEST                                ! SET ERROR FLAG
0117         IF (RTEST) WRITE(5,107)                  ! OUT OF RANGE ERROR
0118

```

```

0119      RTEST=(LNGMAX.LT.LNGMIN)          ! TEST LNG MAX > MIN
0120      R=R+RTEST                          ! SET ERROR FLAG
0121      IF (RTEST) WRITE(5,108)           ! BOUNDARIES ERROR
0122
0123      RTEST=(LNGMAX-LNGMIN.GT.150)       ! TEST RANGE < 150
0124      R=R+RTEST                          ! SET ERROR FLAG
0125      IF (RTEST) WRITE(5,109)           ! RANGE TOO WIDE
0126
0127      RTEST=(LNGMAX-LNGMIN.LT.10)       ! TEST RANGE > 10
0128      R=R+RTEST                          ! SET ERROR FLAG
0129      IF (RTEST) WRITE(5,110)           ! RANGE TOO NARROW
0130
0131      GOTO 200                            ! GO TO ERROR SECTION
0132
0133      !-----ERROE MESSAGE SPECIFICS
0134      150  WRITE(5,151) FNAME             ! FILE NAME WITH ERROR
0135           R=.TRUE.                      ! RESET ERROR FLAG
0136           GOTO 200                      ! SKIP NEXT 2 LINES
0137      152  WRITE(5,153) NDXLEN            ! DATA FILES TOO BIG ERROR
0138           R=.TRUE.                      ! RESET ERROR FLAG
0139      200  RERR=-R                        ! # OF ERRORS IN INDEX FILE
0140           IF (R) WRITE(5,201) RERR       ! DISPLAY # OF ERRORS
0141           IF (R) THEN                    ! ERROR EXISTS
0142               CLOSE (UNIT=4)             ! CLOSE FILE 4
0143               OPEN=.FALSE.               ! SET OPEN FLAG
0144               GO TO 999                  ! RETURN TO CALLING ROUTINE
0145               END IF                     ! END IF BLOCK
0146
0147      !-----TRUNCATE THE LATs, LNGs TO NEAREST FIVE DEGREES
0148           LATMIN=5*LATMIN/5.              ! SET MIN 5 DEGREE LATMIN
0149           LATMAX=5*LATMAX/5.              ! SET MIN 5 DEGREE LATMAX
0150           LNGMIN=5*LNGMIN/5.              ! SET MIN 5 DEGREE LNGMIN
0151           LNGMAX=5*LNGMAX/5.              ! SET MIN 5 DEGREE LNGMAX
0152           J=MAXDTY*(LATMAX-LATMIN)*(LNGMAX-LNGMIN)/25 ! GET # OF INDEXES
0153           IF (J.GT.NDXLEN) GOTO 152       ! CHECK NUMBER OF LOGICAL RECS
0154           RECMAX=(J+4)/64+1              ! GET MAX NUMBER OF RECORDS
0155           W=0                            ! # OF ITEMS READ FROM FILE
0156
0157      !-----GET DATA FROM "MAP@B" FOR THE BASE LAT & LNG
0158           DO 3 REC=1,RECMAX               ! FOR MAX NUMBER OF RECORDS
0159               K=1                         ! INIT ITEM FOR NOT 1ST REC
0160               IF (REC.EQ.1) K=5           ! ACCOUNT FOR LAT;LNG MIN; MAX
0161               READ(4'REC,ERR=4) (RECBUF(I),I=1,64) ! READ INDEX FILE DATA
0162               DO 2 I=K,64                 ! FOR EACH RECORD
0163                   IF (W.LT.J) THEN        ! # OF ITEMS READ < MAXIMUM
0164                       W=W+1               ! INCREMENT # OF RECORDS READ
0165                       RECNDX(W)=RECBUF(I) ! POINTER OF "MAP"(OAC)"A"
0166                   END IF                 ! END IF BLOCK
0167               2  CONTINUE                 ! END DO LOOP
0168               3  CONTINUE                 ! END DO LOOP
0169           GOTO 8
0170
0171      !-----ERROR READING FILE-----
0172      4  WRITE(5,5)                       ! DATA BASE ERROR MESSAGE
0173           DO 6 K=1,I                     ! DO FOR INDEXES THAT DO EXIST
0174               IF (W.GE.J) GO TO 8         ! # OF ITEMS READ >= MAXIMUM
0175               W=W+1                       ! INCREMENT # OF RECORDS READ
0176               RECNDX(W)=RECBUF(I)        ! POINTER OF "MAP"(OAC)"A"
0177      6  CONTINUE                         ! END DO LOOP

```

```

0178          DO 7 K=W,J          ! FOR REMAINDER OF INDEX BLOCK
0179          RECNDX(I)=-TWO15    ! FILL WITH ZEROS
0180      7          CONTINUE      ! END DO LOOP
0181      8          CLOSE (UNIT=4) ! CLOSE DATA FILE
0182          OPEN=.FALSE.        ! SET OPEN ERROR FLAG
0183      999        RETURN        ! RETURN TO CALLING ROUTINE
0184
0185      ! -----FORMAT STATEMENTS-----
0186      5          FORMAT(X,'** WARNING FILE RECORD BUFFER NOT FILLED, PADDING **',
0187          *          /,X,5X,'PROBABLE DATA BASE ERROR')
0188      101        FORMAT(X,'MINIMUM LATITUDE OUT OF RANGE')
0189      102        FORMAT(X,'MAXIMUM LATITUDE OUT OF RANGE')
0190      103        FORMAT(X,'INCONSISTENT LATITUDE BOUNDARIES')
0191      104        FORMAT(X,'LATITUDE RANGE IS TOO NARROW')
0192      105        FORMAT(X,'LATITUDE RANGE IS TOO WIDE')
0193      106        FORMAT(X,'MINIMUM LONGITUDE IS OUT OF RANGE')
0194      107        FORMAT(X,'MAXIMUM LONGITUDE IS OUT OF RANGE')
0195      108        FORMAT(X,'INCONSISTENT LONGITUDE BOUNDARIES (OR IDL CROSSED)')
0196      109        FORMAT(X,'LONGITUDE RANGE IS TOO WIDE')
0197      110        FORMAT(X,'LONGITUDE RANGE IS TOO NARROW')
0198      151        FORMAT(X,'ERROR ENCOUNTERED IN READING DATA FROM ''',21A1,'''')
0199      153        FORMAT(X,'ERROR, OVER 'I3' 5 DEGREE SQUARES IN DATA BASE')
0200      201        FORMAT(X,I2,' ERROR(S) ENCOUNTERED IN INDEX FILE')
0201      END

```

```

0001          SUBROUTINE GETREC(SNDX,PNDX,QUAD,DTYPE,NDX,E)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: GETREC
0005      ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006      ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007      ! DATE: 1982 & 6/84 (FORTRAN 77)
0008      ! FUNCTION: SUBROUTINE GET AND PROCESS LOGICAL RECORD IS
0009      !             DESIGNED TO OBTAIN DATA FOR A 5-DEGREE SQUARE AND CHECK
0010      !             ITS LATITUDE, LONGITUDE, AND DATA TYPE.
0011      ! INPUTS: LOGICAL RECORD
0012      ! OUTPUTS: ERROR MESSAGES TO OPERATOR
0013      ! MODULES CALLED: BMOD
0014      ! CALLED BY: CRUNCH
0015      !
0016      ! =====
0017      ! NOTE: EACH POLYGON READ (IF THERE IS ANY) IS CHECKED FOR VALIDITY.
0018      !       EACH SEGMENT OF THE SQUARE IS ALSO CHECKED FOR VALIDITY.
0019      !       DATA IS THEN ASSIGNED TO A TEMPORARY WORK BUFFER TO BE
0020      !       PROCESSED WITH OTHER QUADRANTS READ FROM THE DATA BASE.
0021      !       ROTATIONAL ANALYSIS IS PERFORMED ON EACH POLYGON TO DETER-
0022      !       MINE THE TOTAL CHANGE OF THE ANGLES BETWEEN EACH POINT IN
0023      !       THE POLYGON, THE POINT OF THE SHIP'S LOCATION, & THE NEXT
0024      !       POINT IN THE POLYGON. THE DISTANCE OF THE CLOSEST POINT
0025      !       IN THE POLYGON IS ASSIGNED AS THE DISTANCE OF THE POLYGON.
0026      ! =====
0027      !
0028          INCLUDE 'MAP.PAR'
0029      1      PARAMETER STOLEN=3800
0030      1      PARAMETER SEGLEN=60, POLLEN=40
0031      1      PARAMETER WRKLEN=1000, NDXLEN=300
0032      1      PARAMETER MAXDTY=3
0033      1      PARAMETER TOL=3
0034      1      PARAMETER DEG=57.2957795
0035      1      PARAMETER RAD=.017453293
0036      1      PARAMETER PI=3.14159265
0037      1      PARAMETER ERAD=3440.3
0038      1      PARAMETER S251=63001
0039      1      PARAMETER TWO15=32768
0040      1
0041      1 !      INTEGER*2 MAXDTY,NDXLEN,POLLEN,SEGLEN,STOLEN,TOL,WRKLEN
0042      1 !      INTEGER*4 S251,TWO15
0043      1 !      REAL*4      DEG,ERAD,PI,RAD
0044          INCLUDE 'CBT.INC'
0045      1 ! -----CBT.INC-----
0046      1 ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0047      1 ! -----
0048      1 ! BTRANS (4,4)      ?
0049      1 !
0050      1      INTEGER*2 BTRANS(4,4)
0051      1
0052      1      COMMON /CBT/  BTRANS
0053      1 ! -----END CBT.INC-----
0054      1
0055          INCLUDE 'CLOC.INC'
0056      1 ! -----CLOC.INC-----
0057      1 ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0058      1 ! -----
0059      1 ! BLAT          BASE LATITUDE                                REAL*4

```

```

0060 1 ! BLNG          BASE LONGITUDE          REAL*4
0061 1 ! LAT           LATITUDE OF SHIP'S LOCATION REAL*4
0062 1 ! LNG           LONGITUDE OF SHIP'S LOCATION REAL*4
0063 1 ! NMLT50        # OF NAUTICAL MILES PER 50TH DEGREE REAL*4
0064 1 !               OF LATITUDE
0065 1 ! NMLG50        # OF NAUTICAL MILES PER 50TH DEGREE REAL*4
0066 1 !               OF LONGITUDE
0067 1 !
0068 1               REAL*4  LAT,LNG,BLAT,BLNG,NMLT50,NMLG50
0069 1
0070 1               COMMON /CLOC/  LAT,LNG,BLAT,BLNG,NMLT50,NMLG50
0071 1 ! -----END CLOC.INC-----
0072 1               INCLUDE 'CLOG.INC'
0073 1 ! -----CLOG.INC-----
0074 1 ! VARBL  SIZE    PURPOSE                                TYPE  RANGE
0075 1 ! -----
0076 1 ! CNVRT(-1:0)
0077 1 ! DG
0078 1 ! DL
0079 1 !
0080 1               BYTE  CNVRT(-1:0),DG,DL
0081 1
0082 1               COMMON /CLOG/  CNVRT,DL,DG
0083 1 ! -----END CLOG.INC-----
0084 1               INCLUDE 'CS.INC'
0085 1 ! -----CS-----
0086 1 ! VARBL  SIZE    PURPOSE                                TYPE  RANGE
0087 1 ! -----
0088 1 ! S      -1,3800  POLYGON AND SEGMENT STORAGE ARRAY  REAL*4
0089 1 ! STOLEN          STORAGE ARRAY LENGTH (FOR SEGS & POLYS) PARM
0090 1 !
0091 1               REAL*4  S(-1:STOLEN)
0092 1
0093 1               COMMON /CS/    S
0094 1 ! -----CS-END-----
0095 1
0096 1
0097 1 ! VARBL  SIZE    PURPOSE                                TYPE  RANGE
0098 1 ! -----
0099 1 ! BDIV          INTERNAL USER FUNCTION NAME          REAL*4
0100 1 ! BMOD          EXTERNAL USER FUNCTION NAME          REAL*4
0101 1 ! CIR           EARTH'S CIRCUMFERENCE                REAL*4
0102 1 ! D            ANALYSIS ALGORITHM                    REAL*4
0103 1 ! DTYPE        DATA TYPE                            INTEGER*2
0104 1 ! E            ERROR FLAG                             BYTE
0105 1 ! I            LOOP COUNTER                            INTEGER*2
0106 1 ! IMAX          MAX LENGTH OF STORAGE ARRAY          INTEGER*2
0107 1 ! J            LOOP COUNTER                            INTEGER*2
0108 1 ! K            LOOP COUNTER                            INTEGER*2
0109 1 ! N            SIGNIFICANT DIGIT TRUNCATION            REAL*4
0110 1 ! NDX           INDEX                                  INTEGER*4
0111 1 ! NPOL          NUMBER OF POLYGONS                     INTEGER*2
0112 1 ! NSEG          NUMBER OF SEGMENTS                     INTEGER*2
0113 1 ! PLEN          POLYGON LENGTH                         INTEGER*2
0114 1 ! PNDX (0:160)  POLYGON INDEX                         INTEGER*2
0115 1 ! PTNDX          STORAGE ARRAY INDEX                   INTEGER*2
0116 1 ! Q            QUOTIENT                                  REAL*4
0117 1 ! QUAD          QUADRANT                                INTEGER*2
0118 1 ! R            ROTATIONAL ANGLE                        REAL*4

```



```

0119      ! REC          RECORD FROM FILE          INTEGER*2
0120      ! RECBUF   (64) RECORD BUFFER          INTEGER*2
0121      ! SNDX    (0:240) SEGMENT INDEX          INTEGER*2
0122      ! T          FACTOR                      REAL*4
0123      ! TEMP      FACTOR                      REAL*4
0124      ! T1         LENGTH OF SEGMENT          REAL*4
0125      ! T2         VALUE OF CURRENT POINT     REAL*4
0126      ! T3         CURRENT POLYGON SEGREFERENCE REAL*4
0127      ! UACOSD     INTERNAL USER FUNCTION NAME REAL*4
0128      ! W          WORK BUFFER INDEX          INTEGER*2
0129      ! WLAT       WORKING LATITUDE           INTEGER*2
0130      ! WLEN       WORK BUFFER LENGTH         INTEGER*2
0131      ! WLNG       WORKING LONGITUDE          INTEGER*2
0132      ! WMAX       WORK BUFFER MAXIMUM        INTEGER*2
0133      ! WRKBUF (1000) WORK BUFFER            INTEGER*2
0134      ! X1         LONGITUDE X FACTOR        REAL*4
0135      ! X2         LONGITUDE X FACTOR        REAL*4
0136      ! X3         LONGITUDE X FACTOR        REAL*4
0137      ! Y1         LATITUDE Y FACTOR         REAL*4
0138      ! Y2         LATITUDE Y FACTOR         REAL*4
0139      ! Y3         LATITUDE Y FACTOR         REAL*4
0140      !
0141      ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0142
0143      INTEGER*4 NDX
0144      INTEGER*2 DTYPE,I,IMAX,J,K,NPOL,NSEG,PLEN,
0145      1          PNDX(0:4*POLLEN),PTNDX,QUAD,REC,RECBUF(64),
0146      2          SNDX(0:4*SEGLN),W,WLAT,WLEN,WLNG,WMAX,
0147      3          WRKBUF(WRKLEN)
0148      REAL*4     CIR,D,N,Q,R,T,TEMP,T1,T2,T3,X1,X2,X3,Y1,Y2,Y3,
0149      1          UACOSD,BMOD,BDIV          ! FUNCTIONS
0150      BYTE      E
0151
0152      UACOSD(D)=DEG*ACOS(D)                ! ARCOSINE FUNC USING DEGREES
0153      BDIV(D,I)=AINT(D/FLOAT(I))           ! TRUNCATING DIVISION(REAL/INT)
0154      CIR=(PI*ERAD)**2                     ! EARTH'S CIRCUMFERENCE
0155      N=10.**5                             ! SIGNIFICANT DIGIT TRUNCATION
0156      WMAX=WRKLEN                          ! WORKING LENGTH
0157      E=.FALSE.                           ! ERROR FLAG
0158
0159      !-----GET WORK BUFFER FROM "MAP"(OAC)"A" FILE
0160      REC=NDX/64+1                          ! DETERMINE INITIAL RECORD
0161      READ (4'REC,ERR=101) RECBUF          ! GET INITIAL RECORD
0162      PLEN=IIFIX(BMOD(FLOATJ(NDX),64)+1.) ! POLYGON LENGTH
0163      WLEN=RECBUF(PLEN)                    ! GET WORK LENGTH
0164      IF (WLEN.GT.WMAX) GOTO 103           ! IF RECORD LEN TOO LONG,ERROR
0165      W=0                                  ! INITIALIZE LENGTH
0166      DO 1 I=1,WMAX                        ! FOR WHOLE OF WORK LENGTH
0167      WRKBUF(I)=0                          ! INITIALIZE WORK BUFFER
0168      1 CONTINUE                          ! END DO LOOP
0169      IF (PLEN.GE.64) GOTO 2               ! POLYGON LENGTH >= 64, SKIP
0170      DO 2 I=PLEN+1,IMIN0(64,WLEN+PLEN) ! GET REST OF FIRST BUFFER
0171      W=W+1                                ! WORK BUFFER INDEX
0172      WRKBUF(W)=RECBUF(I)                 ! FILL REST OF WORK BUFFER
0173      2 CONTINUE                          ! END DO LOOP
0174      DO 4 I=REC+1,REC+(WLEN-W-1)/64+1 ! GET THE REST OF THE BUFFERS
0175      IF (W.GE.WLEN) GOTO 5               ! INDEX < MAX LENGTH
0176      READ (4'I,ERR=101) RECBUF          ! RESD REST OF DATA IN RECORD
0177      DO 3 J=1,IMIN0(64,WLEN-W)         ! ASSIGN ONLY PART OF LAST BUF

```

```

0178          W=W+1          ! INCREMENT WORK BUFFER INDEX
0179          WRKBUF(W)=RECBUF(J) ! FILL WORK BUFFER
0180          CONTINUE      ! END DO LOOP
0181  3
0182  4 CONTINUE
0183  5 IF (WRKBUF(1).NE.DTYPE) GOTO 105! CHECK FOR CORRECT DATA TYPE
0184          ! CHECK FOR CORRECT BASE LAT
0185  IF ((QUAD.EQ.1.OR.QUAD.EQ.2).AND.WRKBUF(2).NE.IIFIX(BLAT)+5) GOT
0186  IF ((QUAD.EQ.3.OR.QUAD.EQ.4).AND.WRKBUF(2).NE.IIFIX(BLAT)) GOTO
0187  WLAT=50*(WRKBUF(2)-IIFIX(BLAT)) ! SET WORKING LAT TO 0 OR 250
0188          ! CHECK FOR CORRECT BASE LNG
0189  IF ((QUAD.EQ.1.OR.QUAD.EQ.3).AND.WRKBUF(3).NE.IIFIX(BLNG)) GOTO
0190  IF ((QUAD.EQ.2.OR.QUAD.EQ.4).AND.WRKBUF(3).NE.IIFIX(BLNG)+5) GOT
0191  WLNG=50*(WRKBUF(3)-IIFIX(BLNG)) ! SET WORKING LNG TO 0 OR 250
0192  NSEG=WRKBUF(4)          ! GET NUMBER OF SEGMENTS
0193  IF (NSEG.LT.1.OR.NSEG.GT.SEGLEN) GOTO 111 ! INVALID # OF SEGS
0194  IF (.NOT.DG) THEN      ! POLYGON DATA
0195      IF (NSEG.LT.4) GOTO 111 ! INVALID # OF SEGMENTS
0196      NPOL=WRKBUF(5)      ! GET NUMBER OF POLYGONS
0197      IF (NPOL.LT.1.OR.NPOL.GT.POLLEN) GOTO 113 ! INVALID NUMBER
0198      END IF              ! END IF BLOCK
0199
0200  !-----STORE NUMBER OF SEGMENTS IN THIS QUADRANT
0201  W=4+CNVRT(DL)          ! WORK BUFFER INDEX
0202  I=IIFIX(S(0))          ! START OF STORAGE ARRAY
0203  IMAX=IIFIX(S(-1))      ! MAX LENGTH OF STORAGE ARRAY
0204  DO 11 J=1+4*CNVRT(DL),NSEG ! FOR NUMBER OF SEGMENTS
0205      T1=FLOATI(WRKBUF(W+1)) ! GET LENGTH OF SEGMENT
0206      IF (T1.LT.2..OR.T1.GT.150.) GOTO 115 ! INVALID SEGEMNT
0207      IF (T1+1.+CNVRT(DG).GT.FLOATI(WMAX-W)) GOTO 117 ! OVERFLOW WOR
0208      IF (T1+4.+CNVRT(DG).GT.FLOATI(IMAX-I)) GOTO 119 ! OVERFLOW STO
0209      S(I+1)=T1          ! ASSIGN LENGTH OF SEGMENT
0210      IF (DG.AND.WRKBUF(W+2).LT.-1) GOTO 121 ! NON-POLYGON
0211      IF (DG) S(I+4)=FLOATI(WRKBUF(W+2)) ! SEG CODE FOR CONTOUR D
0212      DO 10 K=1,IIFIX(T1) ! GET REST OF PTS IN SEGMENT
0213          T2=WRKBUF(W+1+K+CNVRT(DG))+TWO15 ! VALUE OF CURRENT PT
0214          IF (T2.GE.S251) GOTO 123 ! CHECK AND STORE POINT
0215          PTNDX=I+4+CNVRT(DG)+K ! STORAGE ARRAY INDEX
0216          S(PTNDX)=501.*(BDIV(T2,251)+FLOATI(WLAT))+BMOD(T2,251)+FLOAT
0217          IF (K.LE.1) THEN ! FIRST TIME THROUGH
0218              R=0. ! INITIAL DATA FOR ROTATIONAL
0219              D=CIR ! ANALYSIS ALGORITHM;SEE PDIST
0220              X2=(BMOD(S(PTNDX),501)-LNG)*NMLG50 ! LONG X FACTOR
0221              Y2=(BDIV(S(PTNDX),501)-LAT)*NMLT50 ! LAT Y FACTOR
0222              IF (DG) S(I+5)=S(PTNDX) ! UPDATE STORAGE ARRAY
0223          ELSE ! NOT FIRST TIME THROUGH
0224              IF (D.LT.0.25) GOTO 10 ! ANALYSIS ALGORITHM <.25
0225              X1=X2 ! LONGITUDE X FACTOR
0226              Y1=Y2 ! LATITUDE Y FACTOR
0227              X2=(BMOD(S(PTNDX),501)-LNG)*NMLG50 ! LONG X FACTOR
0228              Y2=(BDIV(S(PTNDX),501)-LAT)*NMLT50 ! LAT Y FACTOR
0229              X3=X2-X1 ! LONGITUDE X FACTOR
0230              Y3=Y2-Y1 ! LATITUDE Y FACTOR
0231              IF (X3.EQ.0..AND.Y3.EQ.0.) GOTO 10 ! LAT & LNG = 0
0232              Q=-(X1*X3+Y1*Y3)/(X3**2+Y3**2) ! QUOTIENT
0233              IF (Q.LE.0.) T=X1**2+Y1**2 ! FACTOR
0234              IF (Q.GT.0..AND.Q.LT.1.) T=(X1+Q*X3)**2+(Y1+Q*Y3)**2 ! F
0235              IF (Q.GE.1.) T=X2**2+Y2**2 ! FACTOR
0236              IF (D.GT.T) THEN ! DISTANCE > T
0237                  D=T ! RESET DISTANCE

```

```

0237         IF (DG) S(I+5)=S(PTNDX) ! POLYGON DATA
0238     END IF ! END IF BLOCK
0239     T=SQRT((X1**2+Y1**2)*(X2**2+Y2**2)) ! FACTOR
0240     IF (AINT(T*N)/N.EQ.0.) THEN ! ZERO DISTANCE
0241         D=0. ! SET DISTANCE TO ZERO
0242         R=0. ! SET ROTATION ANGLE TO 0
0243         GOTO 10 ! SKIP NEXT
0244     END IF ! END IF BLOCK
0245     TEMP=UACOSD(AINT(((X1*X2+Y1*Y2)/T)*N)/N) ! FACTOR
0246     IF (X2*Y1.NE.Y2*X1) R=R+SIGN(TEMP,(X2*Y1-Y2*X1)) ! ROT AN
0247     END IF ! END IF
0248 10     CONTINUE ! END DO LOOP
0249     S(I+2)=R ! STORE SUM OF ROT ANGLES
0250     S(I+3)=D ! ASSIGN DIST TO NEAREST POINT
0251     SNDX(J+SNDX(0))=I+1 ! UPDATE SEGMENT INDEX
0252     W=W+IIFIX(T1)+1+CNVRT(DG) ! UPDATE WORK STORAGE BUFF LEN
0253     I=I+IIFIX(T1)+4+CNVRT(DG) ! UPDATE STORAGE ARRAY LENGTH
0254 11     CONTINUE ! END DO LOOP
0255     IF (DG) THEN ! IF NON-POLYGON DATA
0256         SNDX(0)=SNDX(0)+NSEG ! UPDATE LEN OF SEG INDEX ARRAY
0257         GOTO 15 ! SKIP POLYGON PROCESSING
0258     END IF ! END IF BLOCK
0259
0260 !-----STORE POLYGONS OF THIS QUADRANT
0261     DO 14 J=1,NPOL ! FOR ALL POLYS IN SQUARE
0262         T1=FLOATI(WRKBUF(W+2)) ! # OF SEG REFS IN POLYGON
0263         IF (T1.LT.1..OR.T1.GT.25.) GOTO 125! ERROR; BEYOND MAX;MIN
0264         T2=FLOATI(WRKBUF(W+3)) ! # OF LABEL PTS IN POLYGON
0265         IF (T2.LT.0..OR.T2.GT.5.) GOTO 127 ! INVALID; BEYOND MAX/MIN
0266         IF (T1+T2+3..GT.FLOATI(WMAX-W)) GOTO 117 ! ERROR;OVERFLOW WOR
0267         IF (T1+T2+3..GT.FLOATI(IMAX-I)) GOTO 119 ! ERROR; OVERFLOE ST
0268         S(I+1)=FLOATI(WRKBUF(W+1)) ! STORE POLYGON INDEX CO
0269         S(I+2)=T1 ! STORE # OF POLY SEG REFS
0270         S(I+3)=T2 ! STORE # OF LABEL PTS
0271         DO 12 K=4,IIFIX(T1)+3 ! PROCESS POLY SEG REFERENCES
0272             T3=FLOATI(WRKBUF(W+K)) ! CURRENT POLY SEG REFERENCE
0273             IF(ABS(T3).GT.FLOATI(NSEG).OR.T3.GT.-5..AND.T3.LT.1.) GOTO 1
0274             IF(ABS(T3).GT.4.) T3=SIGN(FLOAT(SNDX(IABS(IIFIX(T3)))),T3)
0275             IF(ABS(T3).LE.4.) T3=FLOATI(BTRANS(QUAD,T3)) ! INERIOR BORDE
0276             S(I+K)=T3 ! STORE POLY SEG REFERENCE
0277 12     CONTINUE ! END DO LOOP
0278         DO 13 K=IIFIX(T1)+4,IIFIX(T1+T2)+3 !PROCESS POLYGON LABEL POIN
0279             T3=WRKBUF(W+K)+TWO15 ! STORE LABEL PT FOR CURR POLY
0280             IF (T3.GE.S251) GOTO 131 ! ERROR IN LABEL POINT BRANCH
0281             S(I+K)=501.*(BDIV(T3,251)+FLOATI(WLAT))+BMOD(T3,251)+FLOATI(
0282 13     CONTINUE ! END DO LOOP
0283             PNDX(PNDX(0)+J)=I+1 ! UPDATE POLYGON INDEX COUNT
0284             I=I+IIFIX(T1+T2)+3 ! UPDATE STORAGE ARRAY LENGTH
0285             W=W+IIFIX(T1+T2)+3 ! UPDATE WORK BUFFER LENGTH
0286 14     CONTINUE ! END DO LOOP
0287     PNDX(0)=PNDX(0)+NPOL ! UPDATE POLYGON INDEX
0288 15     S(0)=FLOATI(I) ! STORAGE ARRAY START
0289     GO TO 999 ! RETURN TO CALLING ROUTINE
0290
0291 !-----ERRORS-----
0292 101     WRITE(5,102) ! RECORD LENGTH TOO LONG
0293     GOTO 150 ! GO DISPLAY QUADRANT & RETURN
0294 103     WRITE(5,104) ! END OF FILE ENCOUNTERED
0295     GOTO 150 ! GO DISPLAY QUADRANT & RETURN

```

```

0296      105      WRITE(5,106)          ! WRONG OR MISSING RECORD ID
0297              GOTO 150                ! GO DISPLAY QUADRANT & RETURN
0298      107      WRITE(5,108)          ! WRONG OR MISSING LATITUDE ID
0299              GOTO 150                ! GO DISPLAY QUADRANT & RETURN
0300      109      WRITE(5,110)          ! WRONG OR MISSING LNG ID
0301              GOTO 150                ! GO DISPLAY QUADRANT & RETURN
0302      111      WRITE(5,112)          ! INVALID # OF SEGMENTS
0303              GOTO 150                ! GO DISPLAY QUADRANT & RETURN
0304      113      WRITE(5,114)          ! INVALID NUMBER OF POLYGONS
0305              GOTO 150                ! GO DISPLAY QUADRANT & RETURN
0306      115      WRITE(5,116)          ! INVALID # OF PTS IN SEGMENT
0307              GOTO 150                ! GO DISPLAY QUADRANT & RETURN
0308      117      WRITE(5,118)          ! WORK BUFFER OVERFLOW
0309              GOTO 150                ! GO DISPLAY QUADRANT & RETURN
0310      119      WRITE(5,120)          ! STORAGE ARRAY OVERFLOW
0311              GOTO 150                ! GO DISPLAY QUADRANT & RETURN
0312      121      WRITE(5,122)          ! INVALID SEGMENT CODE
0313              GOTO 150                ! GO DISPLAY QUADRANT & RETURN
0314      123      WRITE(5,124)          ! INVALID POINT LAT/LNG CODE
0315              GOTO 150                ! GO DISPLAY QUADRANT & RETURN
0316      125      WRITE(5,126)          ! INVALID # OF SEGS IN POLYGON
0317              GOTO 150                ! GO DISPLAY QUADRANT & RETURN
0318      127      WRITE(5,128)          ! INVALID # OF LABELS IN POLY
0319              GOTO 150                ! GO DISPLAY QUADRANT & RETURN
0320      129      WRITE(5,130)          ! INVALID SEGMENT REFERENCE
0321              GOTO 150                ! GO DISPLAY QUADRANT & RETURN
0322      131      WRITE(5,132)          ! INVALID LABEL LAT/LNG CODE
0323      150      WRITE(5,151) QUAD,IIFIX(BLAT),IIFIX(BLNG) ! ERROR IN QUADRANT #
0324              WRITE(5,152)          ! PAUSE (RETURN TO CONTINUE)
0325              READ(5,153)            ! READ OPERATOR CONTINUE SIGN
0326      999      RETURN                ! RETURN TO CALLING ROUTINE
0327
0328      !-----FORMAT STATEMENTS-----
0329      102      FORMAT(5X,'SPECIFIED RECORD LENGTH TOO LONG, IN GETREC')
0330      104      FORMAT(5X,'END OF FILE ENCOUNTERED')
0331      106      FORMAT(5X,'WRONG OR MISSING RECORD IDENTIFICATION')
0332      108      FORMAT(5X,'WRONG OR MISSING LATITUDE IDENTIFICATION')
0333      110      FORMAT(5X,'WRONG OR MISSING LONGITUDE IDENTIFICATION')
0334      112      FORMAT(5X,'INVALID NUMBER OF SEGMENTS')
0335      114      FORMAT(5X,'INVALID NUMBER OF POLYGONS')
0336      116      FORMAT(5X,'INVALID NUMBER OF POINTS IN SEGMENT')
0337      118      FORMAT(5X,'WORK BUFFER OVERFLOW')
0338      120      FORMAT(5X,'STORAGE ARRAY OVERFLOW')
0339      122      FORMAT(5X,'INVALID SEGMENT CODE')
0340      124      FORMAT(5X,'INVALID POINT LATITUDE OR LONGITUDE CODE')
0341      126      FORMAT(5X,'INVALID NUMBER OF SEGMENTS IN POLYGON')
0342      128      FORMAT(5X,'INVALID NUMBER OF LABELS IN POLYGON')
0343      130      FORMAT(5X,'INVALID SEGMENT REFERENCE')
0344      132      FORMAT(5X,'INVALID LABEL LATITUDE OR LONGITUDE CODE')
0345      151      FORMAT(5X,'ERROR IN QUADRANT #',I1,' (',I4,',',I4') ! BEL')
0346      152      FORMAT(X,'PAUSE (HIT RETURN TO CONTINUE)')$)
0347      153      FORMAT()
0348      END

```

```

0001          SUBROUTINE GLITCH(NBT,NDLYR)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: GLITCH
0005      ! AUTHOR: STEPHEN LAFLEUR, CODE 3333, NUSC/NLL
0006      ! DATE: 18 SEP 84
0007      ! FUNCTION: REMOVE GLITCHES BELOW THE LAYER
0008      ! INPUTS: PARAMETERS PASSED IN AND VARIABLES IN COMMONS.
0009      ! OUTPUTS:
0010      ! MODULES CALLED: NONE
0011      ! CALLED BY: XBT
0012      !
0013          INCLUDE 'DTV.INC'
0014      1 ! -----DTV-----
0015      1 ! VARBL   SIZE   PURPOSE                                TYPE   RANGE
0016      1 ! -----
0017      1 ! D       (25)   DEPTH                                REAL*4
0018      1 ! DD      (25)   DEPTH                                REAL*4
0019      1 ! NNBT                                NUMBER OF BATHETHERMAL    INTEGER*2
0020      1 ! T       (25)   TEMPERATURE                        REAL*4
0021      1 ! TT      (25)   TEMPERATURE                        REAL*4
0022      1 ! VEL     (25)   VELOCITY                            REAL*4
0023      1 !
0024      1          INTEGER*2 NNBT
0025      1          REAL*4 D,DD,T,TT,VEL
0026      1
0027      1          COMMON /DTV/ D(25),T(25),VEL(25),DD(25),TT(25),NNBT
0028      1 ! -----END DTV-----
0029      !
0030      ! VARBL   SIZE   PURPOSE                                TYPE   RANGE
0031      ! -----
0032      ! I                                COUNTER                                INTEGER*2
0033      ! IP      (25)   POSITIVE OR NON-POSITIVE FLAG                INTEGER*2   0 OR 1
0034      ! L                                COUNTER                                INTEGER*2
0035      ! NBB                                NUMBER OF BT POINTS - 3        INTEGER*2
0036      ! NBT                                NUMBER OF BT POINTS                INTEGER*2
0037      ! NDLYR                                BT LAYER'S POSITION IN ARRAY    INTEGER*2
0038      !
0039      ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0040
0041          INTEGER*2 I,IP,L,NBB,NBT,NDLYR
0042          DIMENSION IP(25)
0043
0044      !-----DETERMINE POSITIVE(1) AND NEGATIVE(-1) GRADIENTS BETWEEN POINTS
0045          DO 200 I=1,NBT-1                                ! FOR FOR # OF BT - 1
0046              IP(I)=1                                      ! POSITIVE GRADIENT
0047              IF(VEL(I+1)-VEL(I).LE.0.) IP(I)=-1          ! NEGATIVE GRADIENT
0048      200          CONTINUE                                ! END DO LOOP
0049
0050      !-----REMOVE ANY GLITCHES BELOW THE LAYER.
0051      !-----THERE ARE NO GLITCHES ABOVE THE LAYER BECAUSE THE LAYER
0052      !-----DEPTH POINT IS THE POINT BEFORE THE FIRST NEGATIVE GRADIENT.
0053
0054          NBB=NBT-3                                ! DETERMINE END OF DO LOOP
0055          DO 255 I=NDLYR,NBB                            ! BEGIN DO LOOP
0056      255          IF(IP(I).NE.IP(I+1).AND.IP(I+1).NE.IP(I+2)) THEN
0057              NBT=NBT-1                                ! REMOVE 3RD POINT IN 4
0058              DO 260 L=I+2,NBT                            ! DO UNTIL # OF BT
0059              D(L)=D(L+1)                                ! DEPTH

```

```

0060          VEL(L)=VEL(L+1)          ! VELOCITY
0061          IP(L)=IP(L+1)            ! GRADIENT
0062 260          CONTINUE              ! END DO LOOP
0063          IP(I+1)=1                ! CALCULATE GRADIENT SIGN
0064          IF(VEL(I+2)-VEL(I+1).LE.0.) IP(I+1)=-1! IN NEW LINE SEG
0065          IF(I.LT.NBB)GOTO 255      ! RECHECK NEW 3RD POINT
0066          END IF                    ! END IF BLOCK
0067 250          CONTINUE              ! END DO LOOP
0068          RETURN                    ! BACK TO CALLING ROUTINE
0069          END                        ! END SUBROUTINE

```

#### COMMAND QUALIFIERS

```
FORTRAN /CHECK=ALL/LIST/SHOW=(INCLUDE,NOMAP) [LAFLEUR]GLITCH.F77
```

```

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)
/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD_LINES /NOCROSS_REFERENCE

```

#### COMPILATION STATISTICS

```

Run Time:          1.14 seconds
Elapsed Time:      1.75 seconds
Page Faults:       314
Dynamic Memory:    126 pages

```

```

0119      ! XMIN          DATA BASE MINIMUM X          REAL*4
0120      ! XOFF          X-AXIS OFFSET                INTEGER*2
0121      ! XSCALE       X PROGRAM SCALE              REAL*4
0122      ! Y            Y COORDINATE                  INTEGER*2
0123      ! YMAX         DATA BASE MAXIMUM Y          REAL*4
0124      ! YMIN         DATA BASE MINIMUM Y          REAL*4
0125      ! YOFF         Y-AXIS OFFSET                INTEGER*2
0126      ! YSCALE       Y PROGRAM SCALE              REAL*4
0127      !
0128      ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0129
0130      REAL*4  A,AX,AY,BDIV,BMOD,SCALE,SRAD,USRCOS,USRSIN,USRTAN
0131      REAL*4  XMIN,XMAX,XSCALE,YMIN,YMAX,YSCALE
0132      INTEGER*2 DTYPE,FNP,I,IX,IY,J,K,L,LEN,M,N
0133      INTEGER*2 SGN,SS,X,XOFF,XVAL,Y,YOFF,YVAL
0134      INTEGER*4 C,HYP,STAR,TITLE(5)
0135      BYTE     BOUND(5),TXT(5),INTER(5),ISTR(5),DIR,LAND,UNMARK
0136      DATA    HYP,STAR  /' - ' , ' * ' /
0137      DATA    BOUND     /'B','o','u','n','d'/
0138      DATA    INTER     /'I','n','t','e','r'/
0139      DATA    SRAD       /60/
0140
0141      !-----FUNCTIONS-----
0142      USRTAN(A)=DEG*TAN(RAD*A)          ! TANGENT FUNCT USING DEG
0143      USRCOS(A)=COS(RAD*A)              ! COSINE FUNCTION USING D
0144      USRSIN(A)=SIN(RAD*A)              ! SINE FUNCTION USING DEG
0145      BDIV(A,I)=AINT(A/FLOAT(I))        ! TRUNCATING DIVISION (RE
0146      IX(A)=NINT((A-XMIN)*XSCALE)+XOFF  ! X SCALE FUNCTION
0147      IY(A)=NINT((A-YMIN)*YSCALE)+YOFF  ! Y SCALE FUNCTION
0148      XVAL(A)=IX(BMOD(A,501))           ! SCALED 501 MODULO X VAL
0149      YVAL(A)=IY(USRTAN(BDIV(A,501)/50+BLAT))! SCALED 501 DIV Y VALUE
0150
0151      !-----INITIALIZE-----
0152      SCALE=360                          ! DISPLAY BOX SCALE (RAST
0153      XOFF=247                           ! X-AXIS OFFSET FOR THE B
0154      YOFF=30                            ! Y-AXIS OFFSET FOR THE B
0155      XMIN=0                             ! DATA BASE MIN X PARAMET
0156      XMAX=500                           ! DATA BASE MAX X PARAMET
0157      YMIN=USRTAN(BLAT)                  ! DATA BASE MIN Y PARAMET
0158      YMAX=USRTAN(BLAT+10)               ! DATA BASE MAX Y PARAMET
0159      YSCALE=SCALE/(YMAX-YMIN)           ! PROGRAM SCALES (10 X 10
0160      XSCALE=SCALE/(XMAX-XMIN)           ! PROGRAM SCALES (10 X 10
0161      CALL INITT(2)                      ! INITIALIZE GRAPHICS MOD
0162
0163      !-----DEFINE PLOTTING AREA BY FRAMING DISPLAY AREA & DRAWIN
0164      CALL MOVE(IX(XMIN),IY(YMIN))       ! MOVE PEN TO THESE COORD
0165      CALL DRAW(IX(XMIN),IY(YMAX))       ! DRAW LINE UP
0166      CALL DRAW(IX(XMAX),IY(YMAX))       ! DRAW LINE TO RIGHT
0167      CALL DRAW(IX(XMAX),IY(YMIN))       ! DRAW LINE DOWN
0168      CALL DRAW(IX(XMIN),IY(YMIN))       ! DRAW LINE TO LEFT
0169      CALL MOVE(IX((XMAX+XMIN)/2),IY(YMIN)) ! MOVE PEN
0170      CALL DRAW(IX((XMAX+XMIN)/2),IY(YMAX)) ! DRAW GRID LINE
0171      CALL MOVE(IX(XMIN),IY(USRTAN(BLAT+5))) ! MOVE PEN
0172      CALL DRAW(IX(XMAX),IY(USRTAN(BLAT+5))) ! DRAW GRID LINE
0173
0174      !-----GET GRAPH TITLE-----
0175      J=0                                ! INITIALIZE INDEX
0176      DO 1 I=(OAC-1)*5+1,OAC*5          ! DO FOR OCEAN NAME ARRAY
0177      J=J+1                              ! INCREMENT INDEX

```

```

0178          TITLE(J)=ONAME(I)                ! SET OCEAN NAME TEXT
0179 1          CONTINUE                        ! END DO LOOP
0180
0181 !-----
0182          SS=3                                ! SET SHIP SIZE (RASTERS)
0183          X=IX(LNG)                          ! X COORDINATE
0184          Y=IY(USRTAN(LAT/50+BLAT))          ! Y COORDINATE
0185          CALL MOVE(X,Y-SS)                  ! MOVE PEN
0186          CALL DRAW(X,Y+SS)                  ! DRAW HALF OF CROSS
0187          CALL MOVE(X-SS,Y)                  ! MOVE PEN
0188          CALL DRAW(X+SS,Y)                  ! FINISH DRAWING CROSS /
0189          DO 2 I=30,360,30                   ! DRAW CIRCLE
0190              CALL DRAW(X+NINT(USRSIN(FLOATI(I))*FLOATI(SS)),
0191 1              Y+NINT(USRCOS(FLOATI(I))*FLOATI(SS)))
0192 2          CONTINUE                        ! END DO LOOP
0193
0194 !-----
0195          AX=(SRAD/NMLG50)*.95                ! SET X COORDINATE
0196          AY=(SRAD/(NMLG50*50.))* .95        ! SET Y COORDINATE
0197          CALL LNTYPE(2)                     ! DOTTED LINE TYPE
0198          CALL MOVE(IX(LNG),IY(USRTAN(LAT/50+BLAT+USRTAN(AY)))) ! MOVE PEN
0199          DO 3 I=5,360,5                     ! DO FOR 360 DEGREES
0200              CALL DRAW(IX(LNG+USRSIN(FLOATI(I))*AX), ! DRAW STEAMING RADIUS
0201 *              IY(USRTAN(LAT/50.+BLAT+USRTAN(AY*USRCOS(FLOATI(I))))))
0202 3          CONTINUE                        ! END DO LOOP
0203          CALL LNTYPE(1)                     ! DRAW DATA TYPE FROM DAT
0204
0205 !-----
0206          IF (DTYPE.LE.2) THEN                ! IF POLYGON DATA
0207              SGN=0                            ! INITIALIZE
0208              DO 25 I=IIFIX(S(0))+1,IIFIX(S(0)+S(IIFIX(S(0)))) ! DRAW ALL PO
0209                  N=IIFIX(S(I))                ! IN THE 10 DEGSQUARE A
0210                  DO 22 J=N+3,N+IIFIX(S(N+1))+2 ! THE ORDER OF THEIR DIST
0211                      K=IABS(IIFIX(S(J)))        ! FROM SHIP'S LOCATION
0212                      IF (K.GT.12.AND.IIFIX(S(K)).GE.2) THEN ! VALID VALUES
0213                          L=FNP(IIFIX(S(J)))      ! LOCATION OF SHIP
0214                          CALL MOVE(XVAL(S(L)),YVAL(S(L))) ! MOVE PEN
0215                          IF (S(J).NE.0.) SGN=IIFIX(SIGN(1.,S(J))) ! SIGN
0216                          DO 21 M=L+SGN,FNP(IIFIX(-S(J))),SGN ! IN ORDER OF DIST
0217                              CALL DRAW(XVAL(S(M)),YVAL(S(M))) ! DRAW POLYGON
0218 21          CONTINUE                        ! END DO LOOP
0219                      S(K)=-S(K)                ! MAKE VALUE NEGATIVE
0220                      END IF                    ! END IF BLOCK
0221 22          CONTINUE                        ! END DO LOOP
0222                  DO 24 J=N+IIFIX(S(N+1))+3,N+IIFIX(S(N+1))+IIFIX(S(N+2))+2 !
0223                      IF (S(N+2).NE.0. .AND. S(N).GT.0.) THEN ! VALID VALUES
0224                          CALL MOVE(XVAL(S(J)),YVAL(S(J))) ! MOVE PEN
0225                          LEN=IIFIX(LOG10(AMAX1(S(N),1.)))+1 ! LENGTH OF STRING
0226                          ENCODE(LEN,23,ISTR) IIFIX(S(N)) ! ENCODE STRING
0227                          CALL TEXT(LEN,ISTR)      ! DRAW LABEL
0228                      END IF                    ! END IF BLOCK
0229 24          CONTINUE                        ! END DO LOOP
0230 25          CONTINUE                        ! END DO LOOP
0231
0232 !-----
0233          ELSE                                ! FOR NON-POLYGON DATA
0234              DO 11 I=IIFIX(S(0))+1,IIFIX(S(0)+S(IIFIX(S(0)))) ! FOR SEG STO
0235                  J=FNP(IIFIX(S(I)))+1          ! SET INDEX
0236                  CALL MOVE(XVAL(S(J)),YVAL(S(J))) ! MOVE PEN

```



```

0237          DO 10 K=J+1,FNP(IIFIX(-S(I)))+1      ! DRAW BOTTOM DEPTH
0238          CALL DRAW(XVAL(S(K)),YVAL(S(K)))      ! CONTOURS
0239 10      CONTINUE                                ! END DO LOOP
0240 11      CONTINUE                                ! END DO LOOP
0241          END IF                                ! END IF BLOCK
0242
0243  !-----LABEL GRAPH'S LATITUDE AND LO
0244          DO 31 I=IIFIX(BLAT),IIFIX(BLAT)+10,5  ! LABEL LAT ON LEFT SIDE
0245          IF (I.GT.0) DIR='N'                    ! NORTH FOR POSITIVE
0246          IF (I.LT.0) DIR='S'                    ! SOUTH FOR NEGATIVE
0247          IF (I.EQ.0) DIR=0                      ! EQUATOR SET TO NULL
0248          CALL MOVE(XOFF-28,IY(USRTAN(FLOAT(I)))) ! MOVE PEN
0249          ENCODE(3,10002,ISTR) IABS(I),DIR      ! ENCODE STRING
0250          CALL TEXT(3,ISTR)                      ! LABEL
0251 31      CONTINUE                                ! END DO LOOP
0252          DO 32 I=IIFIX(BLAT),IIFIX(BLAT)+10,5  ! LABEL LAT ON RIGHT SIDE
0253          IF (I.GT.0) DIR='N'                    ! NORTH FOR POSITIVE
0254          IF (I.LT.0) DIR='S'                    ! SOUTH FOR NEGATIVE
0255          IF (I.EQ.0) DIR=0                      ! EQUATOR SET TO NULL
0256          CALL MOVE(XOFF+374,IY(USRTAN(FLOAT(I)))) ! MOVE PEN
0257          ENCODE(3,10002,ISTR) IABS(I),DIR      ! ENCODE STRING
0258          CALL TEXT(3,ISTR)                      ! LABEL
0259 32      CONTINUE                                ! END DO LOOP
0260          DIR=0                                  ! INITIALIZE DIRECTION
0261          DO 33 I=IIFIX(BLNG),IIFIX(BLNG)+10,5  ! LABEL LONGITUDE
0262          IF (I.GT.0) DIR='E'                    ! EAST FOR POSITIVE
0263          IF (I.LT.0) DIR='W'                    ! WEST FOR NEGATIVE
0264          IF (I.EQ.0) DIR=0                      ! GREENWICH MEAN LINE FOR
0265          CALL MOVE(IX(50*(I-BLNG)-14),IY(USRTAN(BLAT))-14) ! MOVE PEN
0266          ENCODE(4,10001,ISTR) IABS(I),DIR      ! ENCODE STRING
0267          CALL TEXT(4,ISTR)                      ! LABEL AT BOTTOM
0268          CALL MOVE(IX(50*(I-BLNG)-14),IY(USRTAN((BLAT+10)))+14) ! MOVE
0269          ENCODE(4,10001,ISTR) IABS(I),DIR      ! ENCODE TEXT
0270          CALL TEXT(4,ISTR)                      ! LABEL AT TOP
0271 33      CONTINUE                                ! END DO LOOP
0272
0273  !-----TITLE THE PICTURE WITH THE OCEAN ARE
0274          CALL MOVE(20,378)                      ! MOVE PEN
0275          CALL TEXT(20,TITLE(1))                 ! OCEAN AREA NAME
0276          CALL MOVE(39,336)                      ! MOVE PEN
0277          TYPE *,!'!STR /60 NMI Range List/'      ! STEAMING RADIUS
0278          CALL MOVE(23,322)                      ! MOVE PEN
0279          TYPE *,!'!STR /(within dotted circle)/' ! STEAMING RADIUS
0280
0281          IF (DTYPE.EQ.3) THEN                    ! IF BOTTOM DATA
0282          CALL MOVE(28,308)                      ! MOVE PEN
0283          TYPE *,!'!STR /500 Fathom Intervals/' ! DEPTH CONTOUR INTERVALS
0284          END IF                                ! END IF BLOCK
0285          IF (DTYPE.GT.2) GO TO 999              ! IF NON-POLYGON DATA, RE
0286
0287  !-----MAKE A CHART-----
0288          X=48                                    ! X COORDINATE
0289          Y=308                                    ! Y COORDINATE
0290          CALL MOVE(X,Y)                          ! MOVE PEN
0291          TYPE *,!'!STR /Code Distance/'          ! WRITE TITLE
0292          Y=Y-14                                  ! RESET Y COORD
0293          CALL MOVE(X,Y)                          ! MOVE PEN
0294          TYPE *,!'!STR /-----/'                ! UNDERLINE TITLE
0295          Y=Y-14                                  ! RESET Y COORD

```

```

0296      DO 48 I=IIFIX(S(0))+1,IIFIX(S(0)+S(IIFIX(S(0)))) ! DISPLAY OPTIO
0297      J=IIFIX(S(IIFIX(S(I)))) ! SET INDEX CODE
0298      K=IIFIX(S(I+IIFIX(S(IIFIX(S(0)))))) ! SET DISTANCE
0299      IF (K.GT.IIFIX(SRAD)) GOTO 49 ! DISTANCE > RADIUS
0300      IF (J.LT.0) C=STAR ! PROMPT FOR LAND
0301      IF (J.LT.0) LAND=.TRUE. ! DISPLAY LAND TEXT FLAG
0302      IF (J.EQ.0) C=HYP ! PROMPT FOR UNMARKED WAT
0303      IF (J.EQ.0) UNMARK=.TRUE. ! UNMARKED WATER FLAG
0304      IF (J.GT.0) ENCODE(3,10003,C) J ! SET DISTANCE PROMPT
0305      IF (K.LT.0) THEN ! DISTANCE IS NEGATIVE
0306          DO 42 L=1,5 ! SET INTERIOR PROMPT
0307              TXT(L)=INTER(L) ! WRITE 'INTER'
0308      42      CONTINUE ! END DO LOOP
0309      END IF ! END IF BLOCK
0310      IF (K.EQ.0) THEN ! DISTANCE IS ZERO
0311          DO 44 L=1,5 ! SET BOUND PROMPT
0312              TXT(L)=BOUND(L) ! WRITE 'BOUND'
0313      44      CONTINUE ! END DO LOOP
0314      END IF ! END IF BLOCK
0315      IF (K.GT.0) ENCODE(5,46,TXT) K ! SET DISTANCE PROMPT
0316      CALL MOVE(X,Y) ! MOVE PEN
0317      CALL TEXT(3,C) ! DISPLAY CODE
0318      CALL MOVE(X+57,Y) ! MOVE PEN
0319      CALL TEXT(5,TXT) ! DISPLAY DISTANCE TEXT
0320      Y=Y-14 ! RESET Y COORD
0321      48      CONTINUE ! END DO LOOP
0322      49      Y=Y-42 ! SET Y COORDINATE
0323      X=X-32 ! SET X COORDINATE
0324      CALL MOVE(X,Y) ! MOVE PEN
0325      IF (LAND) TYPE *, '!STR /"*" -- Land/' ! LAND FLAG SET
0326      IF (LAND .AND. UNMARK) CALL MOVE(X,Y-14)! MOVE PEN
0327      IF (UNMARK) TYPE *, '!STR /"- " -- Undefined Water/' ! UNMARK
0328      999      RETURN ! RETURN TO CALLING ROUTI
0329
0330      !-----FORMAT STATEMENTS-----
0331      23      FORMAT(I<LEN>)
0332      46      FORMAT(I5)
0333      10001   FORMAT(I3,A1)
0334      10002   FORMAT(I2,A1)
0335      10003   FORMAT(I3)
0336      END

```

```

0001      INTEGER*2 FUNCTION INDX(LAT,LNG,DTYPE)
0002      !
0003      ! PROLOGUE:
0004      ! MODULE NAME: INDX
0005      ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006      ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007      ! DATE: 1982 & 6/84 (FORTRAN 77)
0008      ! FUNCTION: THIS FUNCTION IS DESIGNED TO CALCULATE THE INDEX OF
0009      !             THE POINTER IN RECNDX BY USING THE LATITUDE AND LONGITUDE
0010      !             OF THE FIVE DEGREE SQUARE OF THE DATA TYPE.
0011      ! INPUTS: VARIABLES NEEDED TO CALCULATE INDEX
0012      ! OUTPUTS: THE INDEX OF THE POINTER IN RECNDX
0013      ! MODULES CALLED: NONE
0014      ! CALLED BY: MAP
0015      !
0016      INCLUDE 'CL.INC'
0017 1 ! -----CL.INC-----
0018 1 ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0019 1 ! -----
0020 1 ! LATMAX      MAXIMUM LATITUDE                        INTEGER*2
0021 1 ! LATMIN      MINIMUM LATITUDE                        INTEGER*2
0022 1 ! LNGMAX      MAXIMUM LONGITUDE                        INTEGER*2
0023 1 ! LNGMIN      MINIMUM LONGITUDE                        INTEGER*2
0024 1 !
0025 1      INTEGER*2 LATMIN,LATMAX,LNGMIN,LNGMAX
0026 1
0027 1      COMMON /CL/ LATMIN,LATMAX,LNGMIN,LNGMAX
0028 1 ! -----END CL.INC-----
0029 1
0030      !
0031      ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0032      ! -----
0033      ! DTYPE      CURRENT DATA TYPE                        INTEGER*2
0034      ! ILAT        5 DEG LAT OFFSET FOR OWN SHIP            INTEGER*2
0035      ! ILNG        5 DEG LNG OFFSET FOR OWN SHIP            INTEGER*2
0036      ! LAT         BASE LAT OF THE 5 DEGREE SQUARE          INTEGER*2
0037      ! LNG         BASE LNG OF THE 5 DEGREE SQUARE          INTEGER*2
0038      !
0039      ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0040
0041      INTEGER*2 DTYPE,ILAT,ILNG,LAT,LNG
0042
0043      ILAT=LAT-LATMIN/5      ! 5 DEG LAT OFFSET FOR OWN SHIP POSITION
0044      ILNG=LNG-LNGMIN/5      ! 5 DEG LNG OFFSET FOR OWN SHIP POSITION
0045      INDX=((DTYPE-1)*(LATMAX-LATMIN)/5+ILAT)*(LNGMAX-LNGMIN)/5+ILNG+1
0046      ! INDEX FOR CURRENT TYPE OF 5 DEG SQUARE
0047      RETURN                ! RETURN TO CALLING ROUTINE
0048      END                    ! END SUBROUTINE

```

```

0001          SUBROUTINE INITT(LNSMON)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: INITT
0005      ! AUTHOR: J. CASCIO, W. WACHTER(FORTRAN 77), NUSC/NL, CODE 3333
0006      ! DATE: 1981 & 9/84 (FORTRAN 77)
0007      ! FUNCTION: INITIALIZATION ROUTINE FOR THE TEKTRONIX 4025 TERMINAL
0008      ! INPUTS: NUMBER OF LINES FOR TERMINAL
0009      ! OUTPUTS: INITIALIZED TERMINAL
0010      ! MODULES CALLED: NONE
0011      ! CALLED BY: PLT25, SVPGRF, TIC
0012      ! NOTE: THE TERMINAL ALWAYS KEEPS A BLANK LINE BETWEEN THE MONITOR AND
0013      !       THE WORK SPACE.
0014      !
0015      INCLUDE 'SCREEN.INC'
0016  1 ! -----SCREEN-----
0017  1 !  VARBL  SIZE  PURPOSE                                TYPE  RANGE
0018  1 !  -----
0019  1 !  ICLIP  (4)   CLIP BOUNDARIES                        INTEGER*2
0020  1 !  ISCLIP                CLIPPING FLAG                INTEGER*2  TRUE FALSE
0021  1 !  LENX                LENGTH OF X GRAPHICS BOUNDARY  INTEGER*2
0022  1 !  LENY                LENGTH OF X GRAPHICS BOUNDARY  INTEGER*2
0023  1 !  MAXX                MAXIMUM X GRAPHICS BOUNDARY    INTEGER*2
0024  1 !  MAXY                MAXIMUM Y GRAPHICS BOUNDARY    INTEGER*2
0025  1 !  MINX                MINIMUM X GRAPHICS BOUNDARY    INTEGER*2
0026  1 !  MINY                MINIMUM Y GRAPHICS BOUNDARY    INTEGER*2
0027  1
0028  1          INTEGER*2 ICLIP,LENX,LENY
0029  1          INTEGER*2 MAXX,MAXY,MINX,MINY
0030  1          INTEGER*2 ISCLIP
0031  1
0032  1          COMMON /SCREEN/MINX,MAXX,MINY,MAXY,LENX,LENY,ICLIP(4),ISCLIP
0033  1 ! -----SCREEN END-----
0034      INCLUDE 'USER.INC'
0035  1 ! -----USER-----
0036  1 !  VARBL  SIZE  PURPOSE                                TYPE  RANGE
0037  1 !  -----
0038  1 !  XFCTR                FACTOR FOR X AXIS LENGTH RATIO  REAL*4
0039  1 !  XLEN                LENGTH OF X AXIS                REAL*4
0040  1 !  YFCTR                FACTOR FOR Y AXIS LENGTH RATIO  REAL*4
0041  1 !  YLEN                LENGTH OF Y AXIS                REAL*4
0042  1 !  XMAX                MAXIMUM X COORDINATE              REAL*4
0043  1 !  XMIN                MINIMUM X COORDINATE              REAL*4
0044  1 !  YMAX                MAXIMUM Y COORDINATE              REAL*4
0045  1 !  YMIN                MINIMUM Y COORDINATE              REAL*4
0046  1
0047  1          REAL*4 XMIN,XMAX,YMIN,YMAX,XLEN,YLEN,XFCTR,YFCTR
0048  1
0049  1          COMMON /USER/XMIN,XMAX,YMIN,YMAX,XLEN,YLEN,XFCTR,YFCTR
0050  1 ! -----USER END-----
0051  !
0052  !  VARBL  SIZE  PURPOSE                                TYPE  RANGE
0053  !  -----
0054  !  LNSGRF                # OF LINES IN GRAPGHIC AREA    INTEGER*2
0055  !  LNSMON                # OF LINES USED BY THE TERMINAL  INTEGER*2  2 TO 10
0056  !
0057  ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0058
0059          INTEGER*2 LNSGRF,LNSMON

```

```

0060      DATA MINX,MINY,ICLIP(1),ICLIP(3),MAXX,LENX,ICLIP(2)/4*0,3*639/
0061      DATA ISCLIP/.FALSE./
0062
0063      !-----ESTABLISH WORKSPACE-----
0064      IF (LNSMON.LT.2) LNSMON=2      ! MINIMUM # OF LINES USED BY MON
0065      IF (LNSMON.GT.10) LNSMON=10    ! MAXIMUM # OF LINES USED BY MON
0066      LNSGRF=33-LNSMON              ! # OF LINES IN GRAPHICS WORKSPA
0067      WRITE(5,2) LNSMON,LNSGRF      ! DISPLAY # OF LINES
0068
0069      !-----SCREEN LIMITS FOR THE WORKSPACE-----
0070      MAXY=14*LNSGRF-1              ! MAXIMUM Y IS 433, MINIMUM IS 0
0071      LENY=MAXY                     ! LENGTH OF Y
0072      ICLIP(4)=MAXY                 ! SET FOR CLIPPING AREA
0073      XMIN=FLOATI(MINX)             ! MINIMUM X
0074      XMAX=FLOATI(MAXX)             ! MAXIMUM X
0075      YMIN=FLOATI(MINY)             ! MINIMUM Y
0076      YMAX=FLOATI(MAXY)             ! MAXIMUM Y
0077      XLEN=XMAX-XMIN                ! LENGTH OF X
0078      YLEN=YMAX-YMIN                ! LENGTH OF Y
0079      XFCTR=XLEN/FLOATI(LENX)        ! X FACTOR
0080      YFCTR=YLEN/FLOATI(LENY)        ! Y FACTOR
0081      RETURN                        ! RETURN TO CALLING ROUTINE
0082
0083      !-----FORMAT STATEMENT-----
0084      2      FORMAT(' !MON ',I2,' !GRA 1,',I2)
0085      END

```

```

0001      SUBROUTINE INSERT(N,Z,C,Z0,NU)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: INSERT
0005      ! AUTHOR: G. BROWN & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1974 & 10/83 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE INSERT ALLOWS THE OPERATOR TO INSERT
0008      !              TARGET DEPTH POINT INTO SOUND SPEED PROFILE
0009      ! INPUTS: NEW POINTS TO BE ADDED
0010      ! OUTPUTS: UPDATED ARRAY FOR C AND Z
0011      ! MODULES CALLED: NONE
0012      ! CALLED BY: ACT26,ASIS,DIMUS,ENVIRN,FORCST,NOCONV,OTHERS,PBB18,
0013      !              PBB19,PSSV,RAXIN,SETUP,SIMCZ,SVPGRF,XBT
0014      !
0015      !=====
0016      ! ALGORITHMS USED:
0017      !
0018      !     DEPTH POINT:      Z(N+1) = Z0
0019      !
0020      !     VELOCITY POINT:      C(N+1) = C(N-1) +  $\frac{C(N)-C(N-1)}{Z(N)-Z(N-1)} * (Z0 - Z(N-1))$ 
0021      !
0022      !
0023      !
0024      !     VELOCITY POINT INTERPOLATION:
0025      !
0026      !           C(J) = C(J-1) +  $\frac{C(J+1) - C(J-1)}{Z(J+1) - Z(J-1)} * (Z0 - Z(J-1))$ 
0027      !
0028      !
0029      !=====
0030      !
0031      !  VARBL   SIZE   PURPOSE                                TYPE           RANGE
0032      !  -----
0033      !  C       (1)    VELOCITY                                REAL*4
0034      !  I                                     COUNTER          INTEGER*2
0035      !  J                                     COUNTER          INTEGER*2
0036      !  N                                     # OF DEPTH/VELOCITY PAIRS  INTEGER*2
0037      !  NU                                    SVP INDEX          INTEGER*2
0038      !  NUP                                     UPPER LIMIT INDEX OF TABLE  INTEGER*2
0039      !  Z       (1)    DEPTH                                    REAL*4
0040      !  ZL                                     LOWER RANGE OF INPUT DEPTH  REAL*4
0041      !  ZU                                     UPPER RANGE OF INPUT DEPTH  REAL*4
0042      !  Z0                                     SPECIFIED DEPTH VALUE       REAL*4
0043      !
0044      INTEGER*2 I,J,N,NU,NUP
0045      REAL*4 C,Z,ZL,ZU,Z0
0046      DIMENSION Z(1),C(1)
0047
0048      NU=1
0049      IF (Z0.LT.0.9) GO TO 999
0050      IF (Z0.GT.Z(N)+0.9) THEN
0051          Z(N+1)=Z0
0052          C(N+1)=C(N-1)+(C(N)-C(N-1))/(Z(N)-Z(N-1))*(Z0-Z(N-1))
0053          N=N+1
0054          NU=N
0055          GO TO 999
0056      ELSE
0057          NUP=N
0058          ZL=Z0-0.9
0059          ZU=Z0+0.9

```

```

0060          DO 100 J=2,NUP          ! FIND WHERE TO ADD NEW PAIR
0061          IF (Z(J).GE.ZL) THEN      ! NEW Z ABOVE LOWER RANGE
0062              NU=J                  ! SET # OF PAIRS
0063              IF (Z(J).LE.ZU) GO TO 999 ! TOO CLOSE TO EXISTING POINT
0064              DO 50 I=NUP,J,-1      ! PREPARE TO INSERT NEW POINT
0065                  Z(I+1)=Z(I)        ! MOVE ELEMENTS UP 1
0066                  C(I+1)=C(I)        ! UNTIL REACH PLACE
0067              50 CONTINUE          ! TO INSERT NEW PAIR
0068              Z(J)=Z0                ! NEW PAIR INSERTED AT J
0069              C(J)=C(J-1)+(C(J+1)-C(J-1))/(Z(J+1)-Z(J-1))*(Z0-Z(J-1))
0070              N=N+1                  ! INCREASE # OF PAIRS BY 1
0071              GO TO 999              ! RETURN TO CALLING ROUTINE
0072          END IF                    ! END IF BLOCK
0073          100 CONTINUE              ! END DO LOOP
0074          999 RETURN                ! RETURN TO CALLING ROUTINE
0075          END IF                    ! END IF BLOCK
0076      END                            ! END SUBROUTINE

```

#### COMMAND QUALIFIERS

```

FORTRAN /CHECK=ALL/LIST/SHOW=(INCLUDE,NOMAP) [LAFLEUR]INSERT.F77

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)
/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD_LINES /NOCROSS_REFERENCE

```

#### COMPILATION STATISTICS

```

Run Time:          1.29 seconds
Elapsed Time:      1.68 seconds
Page Faults:       335
Dynamic Memory:    113 pages

```

```

0001      SUBROUTINE INUMBR(IVAL,LENGTH)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: INUMBR
0005      ! AUTHOR: J. CASCIO, W. WACHTER(FORTRAN 77), NUSC/NL, CODE 3333
0006      ! DATE: 1981 & 9/84 (FORTRAN 77)
0007      ! FUNCTION: CONVERT INTEGER*2 VARIABLE INTO ASCII
0008      ! INPUTS: INTEGER*2 VALUE
0009      ! OUTPUTS: ASCII EQUIVALENT
0010      ! MODULES CALLED: TEXT
0011      ! CALLED BY: SVPGRF
0012      !
0013      ! VARBL  SIZE PURPOSE                                TYPE      RANGE
0014      ! -----
0015      ! IVAL      VARIABLE TO BE CONVERTED TO ASCII CODE  INTEGER*2
0016      ! J          LOOP COUNTER                          INTEGER*2
0017      ! LENGTH    NUMBER OF DIGITS IN IVAL                INTEGER*2
0018      ! STRING (10) ASCII CONVERSION OF IVAL              BYTE
0019      !
0020      INTEGER*2 IVAL,J,LENGTH
0021      BYTE      STRING(10)
0022
0023      DO 50 J=1,10                                ! DO FOR ALL ARRAY ELE
0024      STRING(J)=0                                ! ZERO OUT ARRAY
0025      50 CONTINUE                                ! END DO LOOP
0026      IF(LENGTH.LE.0.OR.LENGTH.GT.10) LENGTH=10 ! MAXIMUM LENGTH IS 10
0027      ENCODE(LENGTH,51,STRING) IVAL              ! CONVERT TO ASCII
0028      CALL TEXT(LENGTH,STRING)                   ! ASCII TO BE OUTPUTED
0029      RETURN                                     ! RETURN TO CALLING RO
0030
0031      !-----FORMAT STATEMENT-----
0032      51 FORMAT(I<LENGTH>)
0033      END

```



```

0001          SUBROUTINE KEYPCH(INSSP,NBT,MAPFLG)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: KEYPCH
0005      ! AUTHOR: S. LAFLEUR & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1982 & 12/83 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE KEYPCH ALLOWS THE OPERATOR TO INPUT
0008      !              SPECIFIC PROFILES.
0009      ! INPUTS:  PARAMETERS PASSED IN.  VARIABLES IN COMMONS.
0010      ! OUTPUTS: CRT PROMPTING MESSAGES TO OPERATOR.
0011      ! MODULES CALLED: BT,ICLR,IDATE,KSCAT,LATLNG,SSP,SVPRO
0012      ! CALLED BY: ENVIRN,FORCST
0013      !
0014      ! NOTE: IF CALLED BY ENVIRN (IF IANS = 1)
0015      !         AUTOMATED HISTORICAL SSP MERGED W/ MANUAL BT (IF INSSP = 1)
0016      !         AUTOMATED HISTORICAL SSP (IF INSSP = 2)
0017      ! NOTE: IF CALLED BY FORCST (IF IANS = 2)
0018      !         AUTOMATED HISTORICAL SSP (IF INSSP = 2)
0019      !         MANUAL HISTORICAL SSP (IF INSSP = 3)
0020      !         MANUAL HISTORICAL SSP MERGED WITH MANUAL BT (IF INSSP = 4)
0021      !         OPERATOR WILL KEYPUNCH SPECIFIC PROFILE (IF INSSP = 5)
0022      !
0023      INCLUDE 'DTV.INC'
0024      1 ! -----DTV-----
0025      1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0026      1 ! -----
0027      1 ! D      (25)  DEPTH                                    REAL*4
0028      1 ! DD     (25)  DEPTH                                    REAL*4
0029      1 ! NNBT                                NUMBER OF BATHETHERMAL  INTEGER*2
0030      1 ! T      (25)  TEMPERATURE                             REAL*4
0031      1 ! TT     (25)  TEMPERATURE                             REAL*4
0032      1 ! VEL    (25)  VELOCITY                                REAL*4
0033      1 !
0034      1          INTEGER*2 NNBT
0035      1          REAL*4 D,DD,T,TT,VEL
0036      1
0037      1          COMMON /DTV/  D(25),T(25),VEL(25),DD(25),TT(25),NNBT
0038      1 ! -----END DTV-----
0039      INCLUDE 'ENVN.INC'
0040      1 ! -----ENVN-----
0041      1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0042      1 ! -----
0043      1 ! BIO    (2)  BIOLOGICAL BACK SCATTERING  REAL*4      -57. & -47.
0044      1 ! DLYR                                LAYER DEPTH      REAL*4
0045      1 ! MGS                                MGS PROVINCE    INTEGER*2
0046      1
0047      1          REAL*4  BIO,DLYR
0048      1          INTEGER*2 MGS
0049      1          DATA BIO/-57.,-47./
0050      1
0051      1          COMMON /ENVN/  BIO(2),DLYR,MGS
0052      1
0053      1 ! -----END ENVN-----
0054      INCLUDE 'GRF.INC'
0055      1 ! -----GRF-----
0056      1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0057      1 ! -----
0058      1 ! DBT    (25)  DEPTH OF DEPTH/VEL PAIR      REAL*4
0059      1 ! IANS                                PREDICTION TYPE    INTEGER*2  -2 TO +2

```

```

0060 1 ! ILYR          INDEX FOR LAYER DEPTH          INTEGER*2
0061 1 ! INBT         OPERATOR ENTERED # OF BT POINTS  INTEGER*2
0062 1 ! ISVP         LATEST OR HISTORICAL BT FLAG      INTEGER*2      1 OR 2
0063 1 ! I2000        SVP INDEX FOR 2000 FT DEPTH      INTEGER*2
0064 1 ! VBT          (25) VELOCITY FOR DEPTH PAIR REAL*4      REAL*4
0065 1
0066 1              REAL*4      DBT,VBT
0067 1              INTEGER*2  IANS,ILYR,INBT,ISVP,I2000
0068 1
0069 1              COMMON /GRF/ IANS,ISVP,ILYR,I2000,INBT,DBT(25),VBT(25)
0070 1
0071 1 ! -----END GRF-----
0072              INCLUDE 'LOC.INC'
0073 1 ! -----LOC-----
0074 1 ! VARBL  SIZE  PURPOSE                      TYPE      RANGE
0075 1 ! -----
0076 1 ! INDX          SSP INDEX                    INTEGER*2
0077 1 ! LAT      (4)  LATITUDE                     INTEGER*2
0078 1 ! LONG     (4)  LONGITUDE                     INTEGER*2
0079 1 ! NMAREA  (20)  AREA OCEAN NAME                BYTE
0080 1 ! NOC              NUMBER OF OCEAN             INTEGER*2
0081 1 ! RCZ              RANGE TO CONVERG. ZONE REAL*4
0082 1
0083 1              REAL*4      RCZ
0084 1              INTEGER*2  INDX,LAT, LONG,NOC
0085 1              BYTE      NMAREA(20)
0086 1
0087 1              COMMON /LOC/ LAT(4),LONG(4),NOC,INDX,RCZ,NMAREA
0088 1
0089 1 ! -----END LOC-----
0090              INCLUDE 'SVP.INC'
0091 1 ! -----SVP-----
0092 1 ! VARBL  SIZE  PURPOSE                      TYPE      RANGE
0093 1 ! -----
0094 1 ! BDF          BOTTOM DEPTH IN FATHOMS          REAL*4
0095 1 ! BIOP         BIOLOGICAL BACK SCATTERING COEF  REAL*4
0096 1 ! BTDATE  (9)  DATE OF LAST BT INPUT           BYTE
0097 1 ! BTTIME  (8)  TIME OF LAST BT INPUT           BYTE
0098 1 ! C          (50) VELOCITY (PAIRED WITH Z FOR SVP) REAL*4
0099 1 ! CC        (50) VELOCITY (PAIRED WITH ZZ FOR SVP) REAL*4
0100 1 ! CS          SOUND VELOCITY AT SURFACE        REAL*4
0101 1 ! DEG         TEMPERATURE (DEG)                REAL*4      57.2957795
0102 1 ! EL          LAYER DEPTH                      DATA
0103 1 ! F           FREQUENCY                          REAL*4
0104 1 ! GRDS        GRIDS                            REAL*4      0.0164
0105 1 ! ITO         MINIMAL 2-WAY TRAVEL TIME          INTEGER*2
0106 1 ! MGSOP       MGS PROVINCE NUMBER              INTEGER*2
0107 1 ! N           # OF DEPTH/VELOCITY PAIRS          INTEGER*2
0108 1 ! NN          # OF DEPTH/VELOCITY PAIRS          INTEGER*2
0109 1 ! PI          MATHEMATICAL CONSTANT PI          REAL*4      3.1415927
0110 1 ! SNDATE  (9)  DATE SYS PARMS LAST UPDATED      BYTE
0111 1 ! SNTIME  (8)  TIME SYS PARMS LAST UPDTAED      BYTE
0112 1 ! SYDATE  (9)  CURRENT DATE READ FROM SYSTEM    BYTE
0113 1 ! SYTIME  (8)  CURRENT TIME READ FROM SYSTEM    BYTE
0114 1 ! TMP         TEMPERATURE                      REAL*4
0115 1 ! UMKZ        BOTTOM BACK SCATTERING COEF.        REAL*4      -28.0
0116 1 ! WS          WIND SPEED                        REAL*4
0117 1 ! Z          (50) DEPTH OF POINT OF SOUND SPEED  REAL*4
0118 1 ! ZZ        (50) DEPTH OF POINT OF SOUND SPEED  REAL*4

```

```

0119 1
0120 1      INTEGER*2 ITO,MGSOP,N,NN
0121 1      REAL*4   BDF,BIOP,C(50),CC(50),CS,DEG,EL,F,GRDS
0122 1      REAL*4   PI,TMP,UMKZ,WS,Z(50),ZZ(50)
0123 1      BYTE     SYDATE(9),SYTIME(8),BTDATE(9),BTTIME(8)
0124 1      BYTE     SNDATE(9),SNTIME(8)
0125 1      DATA    PI,DEG,GRDS/3.1415927,57.2957795,0.0164/
0126 1      DATA    UMKZ/-28./
0127 1
0128 1      COMMON /SVP/ F,N,Z,C,EL,MGSOP,BDF,WS,CS,TMP,BIOP,
0129 1          1      UMKZ,PI,DEG,GRDS,ITO,ZZ,CC,NN,
0130 1          2      SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME
0131 1 !-----SVP-END-----
0132 1      INCLUDE 'SVP1.INC'
0133 1 !-----SVP1-----
0134 1 ! VARBL   SIZE      PURPOSE                                TYPE      RANGE
0135 1 ! -----
0136 1 ! BUFFER (224)    HISTORICAL DATA FILE BUFFER          REAL*4
0137 1 ! DS      (30)    HISTORICAL DEPTH                      REAL*4
0138 1 ! J20          # OF DEEP OCEAN DEPTH/VEL PAIRS          INTEGER*2
0139 1 ! NS          TOTAL # OF PAIRS IN HISTORICAL            INTEGER*2
0140 1 ! NSN        MONTH NUMBER (1=JAN.,ETC)                 INTEGER*2   1 TO 12
0141 1 ! SLNTY      SALINITY                                    REAL*4
0142 1 ! VS      (30)    HISTORICAL VELOCITY                  REAL*4
0143 1
0144 1      REAL*4   BUFFER,DS,SLNTY,VS
0145 1      INTEGER*2 J20,NSN,NS
0146 1
0147 1      COMMON /SVP1/ J20,BUFFER(224),NSN,SLNTY,DS(30),VS(30),NS
0148 1 !-----END SVP1-----
0149
0150 ! VARBL   SIZE      PURPOSE                                TYPE      RANGE
0151 ! -----
0152 ! I          COUNTER                                INTEGER*2
0153 ! IBEG      IMONTH ARRAY POINTER                    INTEGER*2
0154 ! IEND      IMONTH ARRAY POINTER                    INTEGER*2
0155 ! IDD       CURRENT DAY                              INTEGER*4
0156 ! INSSP     SSP SELECTED                            INTEGER*2
0157 ! IMONTH (36) MONTH NAMES                            BYTE
0158 ! IMM       CURRENT MONTH                            INTEGER*4
0159 ! IYY       CURRENT YEAR                              INTEGER*4
0160 ! MAPFLG    COUNTER                                INTEGER*2
0161 ! NANS      OPERATOR RESPONDSE                       INTEGER*2
0162 ! NBT       NUMBER OF BT                             INTEGER*2
0163 !
0164 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0165
0166      INTEGER*2 I,IBEG,IEND
0167      INTEGER*2 INSSP,MAPFLG,NANS,NBT
0168      INTEGER*4 IMM,IDD,IYY
0169      BYTE IMONTH(36)
0170      DATA IMONTH/'J','A','N','F','E','B','M','A','R',
0171 1      'A','P','R','M','A','Y','J','U','N',
0172 2      'J','U','L','A','U','G','S','E','P',
0173 3      'O','C','T','N','O','V','D','E','C'/
0174
0175
0176      CALL ICLR                                ! CLEAR SCREEN
0177      IF(INSSP.EQ.5) THEN                      ! KEYPUNCH SSP SELECTED

```

```

0178      WRITE(5,6050)                ! SALINITY PROMPT
0179      READ(5,1370) SLNTY           ! INPUTS SALINITY
0180      CALL ICLR                     ! CLEAR SCREEN
0181      WRITE(5,3006)                ! NAME OCEAN AREA PROMPT
0182      READ(5,3007) NMAREA          ! AREA-NAME FOR LABELING
0183      CALL KSCAT(BIO(1),BIO(2),UMKZ) ! OPERATOR SCAT. COEFF.'S
0184      END IF                        ! END IF BLOCK
0185
0186      70 IF(IANS.NE.2) THEN           ! FORECASTING SELECTED
0187          CALL IDATE(IMM,IDD,IYY)    ! GET CURRENT MONTH
0188          NSN=IMM
0189          IBEG=NSN*3-2               ! ARRAY START POINTER
0190          IEND=IBEG+2                ! ARRAY END POINTER
0191          WRITE(5,1450) (IMONTH(I),I=IBEG,IEND) ! WANT DIFFERENT MONTH?
0192          READ(5,1050) NANS          ! OPERATOR RESPONSE
0193          IF(NANS.NE.'Y') GOTO 80    ! DIFFERENT MONTH WANTED
0194          END IF                    ! END IF BLOCK
0195      CALL ICLR                       ! CLEAR SCREEN
0196      WRITE(5,1245)                  ! MONTH PROMPT
0197      READ(5,1360)NSN                ! OPERATOR INPUTS MONTH
0198      80 IF(NSN.LT.1.OR.NSN.GT.12) GO TO 70 ! GO BACK, INVALID MONTH
0199      IF(.NOT.MAPFLG) GOTO 90        ! NO MAP, SKIP NEXT
0200      IF(IANS.EQ.1) GOTO 100         ! IF OPERATIONAL
0201      IF(INSSP.EQ.2) GOTO 100        ! HISTORICAL SSP
0202      CALL ICLR                       ! CLEAR SCREEN
0203      CALL LATLNG(1)                 ! OPERATOR INPUTS LAT(1)
0204      CALL LATLNG(2)                 ! OPERATOR INPUTS LNG(2)
0205      IF(INSSP.EQ.5) GO TO 90        ! KEYPUNCH, SKIP PROMPT
0206      CALL ICLR                       ! CLEAR SCREEN
0207      WRITE(5,5010)                  ! OCEAN PROMPT
0208      READ(5,1360) NOC               ! OPERATOR INPUTS OCEAN
0209      90 CALL ICLR                    ! CLEAR SCREEN
0210      WRITE(5,1170)                  ! MGS PROVINCE PROMPT
0211      READ(5,1360) MGS               ! OPERATOR INPUTS MGS #
0212      IF(MGS.LT.1.OR.MGS.GT.9) GO TO 90 ! INVALID, TRY AGAIN
0213      IF(INSSP.EQ.5) GO TO 200      ! GO TO CALL SSP
0214      CALL ICLR                       ! CLEAR SCREEN
0215      WRITE(5,5000)                  ! SSP AREA INDEX PROMPT
0216      READ(5,1360) INDX             ! INPUTS SSP AREA INDEX
0217      CALL ICLR                       ! CLEAR SCREEN
0218      WRITE(5,6000)                  ! BOTTOM DEPTH PROMPT
0219      READ(5,1370) BDF               ! INPUTS BOTTOM DEPTH
0220      100 CALL SVPRO(NSN)            ! GET HISTORICAL SSP
0221      IF(INSSP.EQ.1.OR.INSSP.EQ.4) CALL BT(INSSP,NBT) ! INPUTS BT
0222      200 IF(INSSP.EQ.5) CALL SSP(INSSP) ! OPERATOR INPUTS SSP
0223      IF(NANS.EQ.'Y') THEN           ! DIFFERENT MONTH WANTED
0224          IBEG=NSN*3-2               ! SET POINTER
0225          BTDATE(4)=IMONTH(IBEG)     ! RESET BT DATE
0226          BTDATE(5)=IMONTH(IBEG+1)  ! RESET BT DATE
0227          BTDATE(6)=IMONTH(IBEG+2)  ! RESET BT DATE
0228      END IF                        ! END IF BLOCK
0229      CALL ICLR                       ! CLEAR SCREEN
0230      WRITE(5,1400)                  ! WIND SPEED PROMPT
0231      READ(5,1370) WS                ! INPUTS WIND SPEED
0232      MGSOP=MGS                      ! MGS PROVINCE
0233      RETURN
0234
0235      !-----FORMAT STATEMENTS-----
0236      1050      FORMAT(A1)

```

```

0237 1170 FORMAT(/1H$,4X,'****ENTER MGS PROVINCE (1-9)****',T60,' ')
0238 1245 FORMAT(/5X,'****ENTER MONTH YOU WANT PREDICTIONS FOR****',
0239 1 //,8X,'1 = JAN'/8X,'2 = FEB, ETC.',T60,' '$)
0240 1360 FORMAT(I4)
0241 1370 FORMAT(F10.0)
0242 1400 FORMAT(/1H$,4X,'****ENTER TRUE WIND SPEED (XX KTS)****',T60,' ')
0243 1450 FORMAT(' THE CURRENT MONTH IS ',3A1/' DO YOU WANT TO DO'
0244 1 ' PREDICTIONS FOR A DIFFERENT MONTH? ', $)
0245 3006 FORMAT(' ****ENTER AREA NAME****', $)
0246 3007 FORMAT(20A1)
0247 5000 FORMAT(' ENTER SSP INDEX #',T60,' ', $)
0248 5010 FORMAT(// ' 1 = NORTH PACIFIC'//
0249 1 ' 2 = NORTH ATLANTIC'//
0250 2 ' 3 = MEDITERRANEAN'//
0251 3 ' 4 = INDIAN'//
0252 4 ' 5 = NORWEGIAN'////
0253 5 ' ENTER OCEAN AREA CODE',T60,' ', $)
0254 6000 FORMAT(' ENTER BOTTOM DEPTH IN FATHOMS',T60,' ', $)
0255 6050 FORMAT(' ENTER SALINTY (TYPICAL VALUE = 35) ', $)
0256 END

```

```

0001      INTEGER*2 FUNCTION KMOD(I,J)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: KMOD
0005      ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006      ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007      ! DATE: 1982 & 6/84 (FORTRAN 77)
0008      ! FUNCTION: THIS FUNCTION IS DESIGNED TO CALCULATE THE CLOCK
0009      !      ARITHMETIC MODULUS OF THE TWO PARAMETERS PASSED.
0010      ! INPUTS: INTEGER*2S TO BE USED IN CALCULATIONS
0011      ! OUTPUTS: MODULO DIVISOR AND CLOCK ARITHMETIC MODULUS
0012      ! MODULES CALLED: NONE
0013      ! CALLED BY: CNNCT, DNUCT, END1, END2
0014      !
0015      !  VARBL   SIZE      PURPOSE                                TYPE      RANGE
0016      !  -----  -----  -----                                ----      -----
0017      !  I              MODULO DIVISOR                            INTEGER*2
0018      !  J              CLOCK ARITHMETIC MODULUS                    INTEGER*2
0019      !
0020      INTEGER*2 I,J
0021
0022      KMOD=IMOD(I,J)                                ! INTEGER*2 REMAINDER OF I/J
0023      IF (I*J.LT.0) KMOD=J+KMOD                      ! CLOCK ARITHMETIC TOTAL
0024      IF (J.EQ.0) KMOD=0                            ! IF MODULO J IS 0, KMOD IS 0
0025
0026      RETURN                                          ! RETURN TO CALLING ROUTINE
0027      END                                            ! END FUNCTION

```

#### COMMAND QUALIFIERS

```

FORTRAN /CHECK=ALL/LIST/SHOW=(INCLUDE,NOMAP) DBA3:[LAFLEUR]KMOD.F77;1

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)
/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD_LINES /NOCROSS_REFERENCE

```

#### COMPILATION STATISTICS

```

Run Time:          0.48 seconds
Elapsed Time:      1.06 seconds
Page Faults:       306
Dynamic Memory:    105 pages

```

```

0001      SUBROUTINE KSCAT(BIOP1,BIOP2,UMKZ)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: KSCAT
0005      ! AUTHOR: S. LAFLEUR & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 5/84 & 5/84 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE KSCAT ALLOWS THE OPERATOR TO INPUT
0008      !              SCATTERING AND BACK-SCATTERING COEFFICIENTS.
0009      ! INPUTS: HARD COPY SELECTION, OPERATOR INPUTS
0010      ! OUTPUTS: CRT PROMPTING MESSAGES TO OPERATOR
0011      ! MODULES CALLED: ICLR
0012      ! CALLED BY: KEYPCH
0013      !
0014      !  VARBL   SIZE      PURPOSE                                TYPE      RANGE
0015      !  -----
0016      !  BIOP1      DAY BIO SCATTERING COEFFICIENT      REAL*4      -57
0017      !  BIOP2      NIGHT BIO SCATTERING COEFF          REAL*4      -47
0018      !  JANS       OPERATOR RESPONSE                  INTEGER*2
0019      !  UMKZ       BOTTOM BACK-SCATTERING COEFF        REAL*4
0020      !
0021      INTEGER*2 JANS
0022      REAL*4 BIOP1,BIOP2,UMKZ
0023
0024      BIOP1= -57.          ! DAY BIO SCATTERING COEFF.
0025      BIOP2= -47.          ! NIGHT BIO SCATTERING COEFF.
0026      UMKZ= -28.          ! BOTTOM BACK-SCATTERING COEFF.
0027      CALL ICLR           ! CLEAR SCREEN
0028      WRITE(5,4000)       ! BIO SCATTERING COEFF. TITLE
0029      WRITE(5,4007)       ! PROMPT SELECTION
0030      READ(5,1360) JANS    ! OPERATOR SELECTION
0031      IF(JANS.NE.0) THEN  ! OPERATOR INPUT BIO. SCAT.
0032          WRITE(5,4010)   ! DAY BIO SCAT. COEFF. PROMPT
0033          READ(5,1370) BIOP1 ! INPUTS DAY BIO. SCAT. COEFF.
0034          WRITE(5,4020)   ! NIGHT BIO SCAT. COEFF. PROMPT
0035          READ(5,1370) BIOP2 ! INPUTS NIGHT BIO. SCAT. COEFF.
0036          END IF         ! END IF BLOCK
0037      CALL ICLR           ! CLEAR SCREEN
0038      WRITE(5,4003)       ! BOTTOM BIO SCAT. COEFF. PROMPT
0039      WRITE(5,4007)       ! SELECTION PROMPT
0040      READ(5,1360) JANS    ! OPERATOR SELECTION
0041      IF(JANS.EQ.0) THEN  ! OPERATOR INPUT BOT. BACK-SCAT.
0042          WRITE(5,4030)   ! BOT. BACK-SCAT. COEFF. PROMPT
0043          READ(5,1370) UMKZ ! INPUTS BOT. BACK-SCAT. COEFF.
0044          END IF         ! END IF BLOCK
0045      CALL ICLR           ! CLEAR SCREEN
0046      RETURN              ! RETURN TO CALLING ROUTINE
0047
0048      !-----FORMAT STATEMENTS-----
0049      1360      FORMAT(I4)
0050      1370      FORMAT(F10.0)
0051      4000      FORMAT(' BIO SCATTERING COEFF.S ENTRY')
0052      4003      FORMAT(' BOTTOM BACK-SCATTERING COEFF.S ENTRY')
0053      4007      FORMAT('// ' 0 = USE DEFAULT VALUES'/
0054      1          ' ' 1 = OPERATOR INPUT'//
0055      2          '*** ENTER YOUR CHOICE ***',T58,$)
0056      4010      FORMAT(' ENTER DAYTIME BIO SCATTERING COEFF.',T58,$)
0057      4020      FORMAT(' ENTER NIGHTTIME BIO SCATTERING COEFF.',T58,$)
0058      4030      FORMAT(' ENTER BOTTOM BACK-SCATTERING COEFF.',T58,$)
0059      END

```

```

0001          SUBROUTINE LATLNG(OPT)
0002 )002
0003 ! PROLOGUE:
0004 ! MODULE NAME: LATLNG
0005 ! AUTHOR: S. LAFLEUR & W. WACHTER, CODE 3333, NUSC/NLL
0006 ! DATE: 1982 & 12/83 (FORTRAN 77)
0007 ! FUNCTION: SUBROUTINE LATLNG PRODUCES EITHER THE LATITUDE
0008 !            OR LONGITUDE FOR THE SHIP'S LOCATION. OPTION 1 IS
0009 !            FOR LATITUDE. OPTION 2 IS FOR LONGITUDE.
0010 ! INPUTS:  PARAMETERS PASSED IN. VARIABLES IN COMMONS. SEE NOTE.
0011 ! OUTPUTS: CRT PROMPTING MESSAGES TO OPERATOR.
0012 ! MODULES CALLED: NONE
0013 ! CALLED BY: KEYPCH
0014 !
0015 ! -----
0016 ! NOTE:  THE USER INPUT IS A BUFFER OF UP TO 12 CHARACTERS LONG WITH
0017 !        EACH PARAMETER SEPARATED BY EITHER COMMAS OR BLANKS.
0018 !        THE FORMAT IS <SDDD><MM><CC><R> WHERE:
0019 !                S=THE SIGN (EITHER '=', '+', OR NOTHING)
0020 !                DEFAULT: '+' UNLESS 'S' OR 'W' SET
0021 !                D=THE DEGREES (0 = 80 FOR LAT) (0 = 180 FOR LONG)
0022 !                M=THE MINUTES (0 = 59)
0023 !                C=THE SECONDS (0 = 59)
0024 !                R=THE DIRECTION (EITHER 'S', 'N', 'E', 'W' OR NOTHING)
0025 !                DEFAULT: 'N' OR 'E' UNLESS '=' SET
0026 ! -----
0027 !
0028          INCLUDE 'LOC.INC'
0029 1 ! -----LOC-----
0030 1 ! VARBL  SIZE  PURPOSE  TYPE  RANGE
0031 1 ! -----
0032 1 ! INDX  SSP INDEX  INTEGER*2
0033 1 ! LAT   (4)  LATITUDE  INTEGER*2
0034 1 ! LONG  (4)  LONGITUDE  INTEGER*2
0035 1 ! NMAREA (20) AREA OCEAN NAME  BYTE
0036 1 ! NOC   NUMBER OF OCEAN  INTEGER*2
0037 1 ! RCZ   RANGE TO CONVERG. ZONE REAL*4
0038 1
0039 1          REAL*4    RCZ
0040 1          INTEGER*2 INDX,LAT, LONG, NOC
0041 1          BYTE      NMAREA(20)
0042 1
0043 1          COMMON /LOC/ LAT(4), LONG(4), NOC, INDX, RCZ, NMAREA
0044 1
0045 1 ! -----END LOC-----
0046 1
0047 1 ! VARBL  SIZE  PURPOSE  TYPE  RANGE
0048 1 ! -----
0049 1 ! BUF   (12)  BUFFER PASSED IN  BYTE
0050 1 ! FLAG  (3)  ERROR FLAG  BYTE  TRUE/FALSE
0051 1 ! I     COUNTER  INTEGER*2
0052 1 ! J     COUNTER  INTEGER*2
0053 1 ! LEN   LENGTH OF BUFFER  INTEGER*2
0054 1 ! MAXVAL (3)  MAXIMUM VALUE OF T  INTEGER*2 180,59,59
0055 1 ! MINUS  HEMISPHERE FLAG  BYTE  TRUE/FALSE
0056 1 ! NSEW  (2,2) HEMISPHERE 'S','N','W','E'  BYTE  S,N,W,E
0057 1 ! NUMBER FLAG FOR EXISTANCE OF NUMBERS  BYTE  TRUE/FALSE
0058 1 ! ONE   FLAG FOR OPTION 1 (LATITUDE)  BYTE  TRUE/FALSE
0059 1 ! OPT   LATITUDE(1) OR LONGITUDE(2) OPTION  INTEGER*2 1 OR 2

```



```

0060 ! SIGN          FLAG FOR PLUS OR MINUS SIGN          BYTE      TRUE/FALSE
0061 ! SPEC      (3)  ' DEGREE', ' MINUTE', ' SECOND' ARRAY REAL*8
0062 ! T          (3)  NUMERICAL SPECIFICATION OF SPEC    INTEGER*2
0063 ! TWO        FLAG FOR OPTION 1 (LATITUDE)            BYTE      TRUE/FALSE
0064 ! W          BUFFER POINTER                          INTEGER*2
0065 !
0066 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMON ***
0067
0068          INTEGER*2 I,J,LEN,MAXVAL(3),OPT,T(3),W
0069          REAL*8   SPEC(3)
0070          BYTE     BUF(12),NSEW(2,2),ONE,TWO,FLAG(3),MINUS,SIGN,NUMBER
0071          DATA    NSEW/'S','N','W','E'//,
0072          1        SPEC/' DEGREE',' MINUTE',' SECOND'//,
0073          2        MAXVAL/180,59,59/
0074
0075 !=====PRELIMINARIES=====
0076          ONE=(OPT.EQ.1)          ! SET OPTION 1 FLAG (LAT)
0077          TWO=(OPT.EQ.2)          ! SET OPTION 2 FLAG (LONG)
0078 1      T(1)=0                    ! INITIALIZE DEGREES
0079          T(2)=0                    ! INITIALIZE MINUTES
0080          T(3)=0                    ! INITIALIZE SECONDS
0081          FLAG(1)=.FALSE.         ! RESET ERROR FLAGS
0082          FLAG(2)=.FALSE.         ! RESET ERROR FLAGS
0083          FLAG(3)=.FALSE.         ! RESET ERROR FLAGS
0084          NUMBER=.FALSE.          ! RESET NUMBER FLAG
0085
0086 !-----GET DATA STRING (LAT OR LONG)-----
0087          IF (ONE) THEN            ! OPTION 1
0088              WRITE(5,2)           ! INPUT LATITUDE
0089          ELSE                      ! OPTION 2
0090              WRITE(5,4)           ! INPUT LONGITUDE
0091          END IF                   ! END IF BLOCK
0092          READ(5,6) LEN,BUF        ! GET BUFFER
0093          IF (LEN.EQ.0) GOTO 101    ! ILLEGAL ENTRY ERROR
0094
0095 !=====PARSE THE INPUT STRING=====
0096          DO 7 I=1,LEN             ! CHECK FOR ALL BLANKS
0097              IF (BUF(I).NE.' ') GOTO 8 ! NON-BLANK FOUND
0098 7      CONTINUE                  ! END DO LOOP
0099          GOTO 101                ! ILLEGAL ENTRY ERROR
0100
0101 !-----PARSE + OR - SIGN (IF PRESENT)-----
0102 8      W=0                       ! SET BEGINNING OF THE BUF
0103          MINUS=.FALSE.           ! DEFAULT POSITIVE HEMISPHERE
0104          SIGN=.FALSE.            ! DEFAULT NO SIGN PRESENT
0105          IF (BUF(W+1).NE.'+' .AND. BUF(W+1).NE.'-') GOTO 9 ! NO SIGN
0106          W=1                     ! GET NEXT CHAR
0107          MINUS=(BUF(W).EQ.'-')    ! SET FLAG ACCORDING TO + -
0108          SIGN=.TRUE.              ! THERE IS A SIGN
0109          IF (W.EQ.LEN) GOTO 103   ! NOT LAT OR LONG INPUT ERROR
0110
0111 !-----PARSE DEGREE, MINUTE, & SECOND #S AND/OR DELIMITERS-----
0112 9      DO 12 I=1,3               ! DO THREE TIMES
0113          DO 10 J=1,4             ! DO FOUR TIMES
0114              W=W+1               ! CHECK FOR DELIMITERS OR #S
0115              IF (BUF(W).EQ.' ' .OR. BUF(W).EQ.',') GOTO 11 ! ERROR
0116              IF (BUF(W).LT.'0' .OR. BUF(W).GT.'9') GOTO 13 ! ERROR
0117              T(I)=T(I)*10+(BUF(W)-48) ! ASCII TO INTEGER*2
0118              NUMBER=.TRUE.         ! FLAG A LEAST ONE NUMBER SET

```

```

0119      10          CONTINUE                                ! END DO LOOP
0120      11          FLAG(I)=(T(I).GT.MAXVAL(I))              ! SET FLAG FOR ERROR
0121      12          CONTINUE                                  ! END DO LOOP
0122
0123      !-----PARSE OFF DELIMITER FOR HEMISPHERE (IF ANY)
0124      13          IF (.NOT.NUMBER) GOTO 103                  ! CHECK TO SEE IF ANY # SET
0125      14          IF (W.GT.LEN) GOTO 16                      ! CHECK CURRENT LENGTH
0126          IF (BUF(W).NE.' '.AND.BUF(W).NE.',') GOTO 15 ! NO HEMISPHERE
0127          W=W+1                                              ! SKIP DELIMITERS
0128          GOTO 14                                            ! START AGAIN
0129
0130      !-----PARSE, CHECK, AND SET HEMISPHERE (N,S,E,OR W)
0131      15          IF (BUF(W).NE.NSEW(1,OPT) .AND. BUF(W).NE.NSEW(2,OPT))
0132          1          GOTO 101                                ! ILLEGAL ENTRY ERROR
0133          IF (SIGN .AND. (MINUS .XOR. (BUF(W).EQ.NSEW(1,OPT))))
0134          1          GOTO 105                                ! SIGN CONFLICT ERROR
0135          MINUS=(BUF(W).EQ.NSEW(1,OPT))                      ! SET HEMISPHERE SIGN
0136
0137      !-----OUTPUT ERRORS OF DEGREES, MINUTES, SECONDS
0138      16          DO 18 I=1,3                                ! PRINT OUT SPEC ERRORS
0139          IF (FLAG(I)) THEN                                  ! ERROR FLAG
0140              WRITE(5,17) SPEC(I)                            ! ERROR - ILLEGAL SPEC
0141              WRITE(5,1002)                                    ! TRY AGAIN
0142              FLAG(1)=FLAG(1)+FLAG(I)                        ! RESET FLAG
0143          END IF                                              ! END IF BLOCK
0144      18          CONTINUE                                    ! END DO LOOP
0145          IF (FLAG(1)) GOTO 1                                ! IF ANY FLAG SET, TRY AGAIN
0146
0147      !=====SET FINAL LATITUDE OR LONGITUDE PARAMETERS
0148          SIGN=2*MINUS+1                                      ! SET SIGN OF LAT OR LNG
0149          IF (ONE) THEN                                       ! OPTION 1
0150              LAT(1)=T(1)                                     ! DEGREES LATITUDE FOR SIMAS
0151              LAT(2)=T(2)                                     ! MINUTES LATITUDE FOR SIMAS
0152              LAT(3)=T(3)                                     ! SECONDS LATITUDE FOR SIMAS
0153              LAT(4)='S'                                     ! SOUTH
0154              IF(SIGN.EQ.1) LAT(4)='N'                        ! NORTH
0155          ELSE                                                ! OPTION 2
0156              LONG(1)=T(1)                                    ! DEGREES LONGITUDE FOR SIMAS
0157              LONG(2)=T(2)                                    ! MINUTES LONGITUDE FOR SIMAS
0158              LONG(3)=T(3)                                    ! SECONDS LONGITUDE FOR SIMAS
0159              LONG(4)='W'                                     ! WEST
0160              IF(SIGN.EQ.1) LONG(4)='E'                       ! EAST
0161          END IF                                              ! END IF BLOCK
0162          RETURN                                              ! RETURN TO CALLING ROUTINE
0163
0164      !=====ERRORS=====
0165      101          WRITE(5,102)                               ! ILLEGAL ENTRY ERROR
0166          WRITE(5,1002)                                       ! TRY AGAIN
0167          GOTO 1                                              ! START OVER
0168      103          WRITE(5,104)                               ! NOT LAT OR LONG INPUT ERROR
0169          WRITE(5,1002)                                       ! TRY AGAIN
0170          GOTO 1                                              ! START OVER
0171      105          WRITE(5,106)                               ! SIGN CONFLICT ERROR
0172          WRITE(5,1002)                                       ! TRY AGAIN
0173          GOTO 1                                              ! START OVER
0174
0175      !=====FORMAT STATEMENTS=====
0176      2          FORMAT(X,'ENTER LATITUDE <DEG>,<MIN>,<SEC>,<'N' OR 'S'> '$')
0177      4          FORMAT(X,'ENTER LONGITUDE <DEG>,<MIN>,<SEC>,<'E' OR 'W'> '$')

```

```

0178      6      FORMAT(Q,12A1)
)179      17      FORMAT(X,'*** ERROR, ILLEGAL',A8,'SPECIFICATION'$)
0180      102      FORMAT(X,'*** ERROR, ILLEGAL ENTRY'$)
0181      104      FORMAT(X,'*** ERROR, NO LONGITUDE OR LATITUDE INPUT'$)
0182      106      FORMAT(X,'*** ERROR, CONFLICTING DEGREES SIGN AND HEMISPHERE'$)
0183      1002      FORMAT('+ , TRY AGAIN ***')
0184      END

```

#### COMMAND QUALIFIERS

```

FORTRAN /CHECK=ALL/LIST/SHOW=(INCLUDE,NOMAP) [LAFLEUR]LATLNG.F77

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)
/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD_LINES /NOCROSS_REFERENCE

```

#### COMPILATION STATISTICS

```

Run Time:          3.24 seconds
Elapsed Time:      9.32 seconds
Page Faults:       385
Dynamic Memory:    152 pages

```

```

0001      SUBROUTINE LAYER(N,Z,C,DLYR)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: LAYER
0005      ! AUTHOR: G. BROWN & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1974 & 12/83 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE LAYER EXAMINES THE SOUND VELOCITY PROFILE
0008      !              DATA ARRAYS TO DETERMINE THE DEPTH OF THE SURFACE
0009      !              DUCT LAYER.
0010      ! INPUTS:  PARAMETERS PASSED IN: C,N,Z
0011      ! OUTPUTS: PARAMETER PASSED OUT: DLYR
0012      ! MODULES CALLED: NONE
0013      ! CALLED BY: ENVIRN, FORCST, XBT
0014      !
0015      !  VARBL   SIZE      PURPOSE                                TYPE      RANGE
0016      !  -----
0017      !  C       (50)      VELOCITY                                REAL*4
0018      !  DLYR                                LAYER DEPTH          REAL*4
0019      !  J                                COUNTER                INTEGER*2
0020      !  N                                INDEX                   INTEGER*2
0021      !  Z       (50)      DEPTH                                    REAL*4
0022      !
0023      INTEGER*2 J,N
0024      REAL*4    C,DLYR,Z
0025      DIMENSION Z(50),C(50)
0026
0027      DO 400 J=2,N
0028      IF(C(J).LT.C(J-1)) GO TO 500
0029      400 CONTINUE
0030      J=N+1
0031      500 DLYR=Z(J-1)
0032
0033      RETURN
0034      END

```

! DO FROM 2 TO INDEX  
! VELOCITY < PREVIOUS  
! END DO LOOP  
! SET COUNTER  
! LAYER DEPTH = DEPTH(J-1)  
! RETURN TO CALLING ROUTINE  
! END SUBROUTINE

18-Dec-1984 13:00:03  
18-Dec-1984 13:00:02

# SUBROUTINE LEROY(ZF,TF,S,V)

```

0001
0002
0003 | PROLOGUE:
0004 | MODULE NAME: LEROY
0005 | AUTHOR: G. BROWN & W. WACHTER, CODE 3333, NUSC/NLL
0006 | DATE: 1974 & 12/83 (FORTRAN 77)
0007 | FUNCTION: SUBROUTINE LEROY CONVERTS A DEPTH/TEMPERATURE PAIR
0008 |           INTO A DEPTH/SOUND VELOCITY PAIR.
0009 | INPUTS:  PARAMETERS PASSED IN: ZF,TF,S
0010 | OUTPUTS: PARAMETERS PASSED OUT: V
0011 | MODULES CALLED: NONE
0012 | CALLED BY: METRIC,VELTMP,XBT
0013 |
0014 | =====
0015 | ALGORITHMS USED:
0016 |
0017 |     SOUND VELOCITY IN FEET/SEC = SOUND VELOCITY IN METERS/SEC
0018 |                               DIVIDED BY 0.3048006
0019 | =====
0020 |
0021 | VARBL  SIZE  PURPOSE                                TYPE  RANGE
0022 | -----
0023 | FTOM    FEET TO METERS CONVERSION  REAL*4  0.3048006
0024 | PSI     PRESSURE KG/CM**2          REAL*4
0025 | S       SALINITY                   REAL*4
0026 | T       TEMPERATURE CENTIGRADE     REAL*4
0027 | TF      TEMPERATURE FARENHITE      REAL*4
0028 | V       VELOCITY                   REAL*4
0029 | VA      SOUND VEL FROM SALINITY     REAL*4
0030 | VB      SOUND VEL FROM PRESSURE     REAL*4
0031 | VY
0032 | VZ      REAL*4
0033 | VO      SOUND VEL FROM TEMPERATURE REAL*4
0034 | Z       DEPTH IN METERS            REAL*4
0035 | ZF      DEPTH IN FEET              REAL*4
0036 |
0037 | REAL*4 FTOM,PSI,S,T,TF,V,VA,VB,VY,VZ,V0,Z,ZF
0038 | DATA FTOM/0.3048006/
0039 |
0040 | Z=FTOM*ZF           ! CONVERT INPUT DEPTH TO METERS
0041 | PSI=0.001*Z         ! PRESSURE KG/CM**2
0042 | T=5.*(TF-32.)/9.    ! CONVERT TEMPERATURE TO CENTIGRADE
0043 | VY=(3.-0.006*(T-10.))
0044 | VZ= (4.*(T-18.)+(S-35.))
0045 | V0=1492.9+(T-10.)*VY-0.01*(T-18.)*VZ+1.2*(S-35.)+Z/61
0046 | V0=1492.9+(T-10.)*(3.-0.006*(T-10.))-0.01*(T-18.)* ! SOUND VEL:
0047 | 1 (4.*(T-18.)+(S-35.))+1.2*(S-35.)+Z/61           ! FROM SALINITY
0048 | VA=0.1*PSI*(PSI+0.5)+0.0002*(PSI*(T-18.))*2 ! FROM PRESSURE
0049 | VB=2.E-7*T*(T-10.)*4 ! FROM TEMPURTURE
0050 | V=(V0+VA+VB)/FTOM   ! CONVERT FROM METERS/SEC TO FEET/SEC
0051 |
0052 | RETURN              ! RETURN TO CALLING ROUTINE
0053 | END                 ! END SUBROUTINE

```

C-34.1

```

0001      SUBROUTINE LNTYPE(LINTYP)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: LNTYPE
0005      ! AUTHOR: J. CASCIO, W. WACHTER(FORTRAN 77), NUSC/NL, CODE 3333
0006      ! DATE: 1981 & 9/84 (FORTRAN 77)
0007      ! FUNCTION: SETS THE LINE TYPE FOR DRAWING OR ERASING LINES
0008      ! INPUTS: TYPE TO SET LINE TYPE TO
0009      ! OUTPUTS: SET LINE TYPE
0010      ! MODULES CALLED: NONE
0011      ! CALLED BY: PLT25, SVPGRF
0012      !
0013      ! VARBL   SIZE PURPOSE                                TYPE      RANGE
0014      ! -----
0015      ! ITYPE      POINTER FOR LINTYP ARRAY                INTEGER*2
0016      ! LINTYP (10) ARRAY OF TYPES OF LINES AVAILABLE      INTEGER*2
0017      !
0018      INTEGER*2 LINTYP, ITYPE
0019      DIMENSION ITYPE(10)
0020      DATA ITYPE/'1','2','3','4','5','6','7','8','P','E'/
0021
0022      IF (LINTYP.LT.1 .OR. LINTYP.GT.10) LINTYP=1 ! INVALID, SET TO SO
0023      WRITE(5,1) ITYPE(LINTYP) ! SET LINE TYPE
0024      RETURN ! RETURN TO CALLING ROUTINE
0025
0026      !-----FORMAT STATEMENT-----
0027      1 FORMAT(' !LIN ',A1)
0028      END

```

```

0001      SUBROUTINE LYRMOD(NBT,DLYR,NDLYR,HLYR)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: LYRMOD
0005      ! AUTHOR: S. LAFLEUR, W. WACHTER (FORTRAN 77)
0006      ! DATE: 7/84 & 7/84 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE LYRMOD MODIFIES THE BT LAYER DEPTH TO
0008      !              BE PLUS OR MINUS 50 FEET OF THE HISTORICAL LAYER DEPTH,
0009      !              WHEN THE BT LAYER DEPTH IS NOT < 100 FEET OR THE BT LAYER
0010      !              DEPTH IS NOT <= THE HISTORICAL LAYER DEPTH.
0011      ! INPUTS: PARAMETERS PASSED IN & VARIABLES IN COMMONS.
0012      ! OUTPUTS: MODIFIED LAYER DEPTH AND SS AT LAYER DEPTH
0013      ! MODULES CALLED: NONE
0014      ! CALLED BY: XBT
0015      !
0016      INCLUDE 'DTV.INC'
0017 1 !-----DTV-----
0018 1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0019 1 ! -----
0020 1 ! D      (25)  DEPTH                                REAL*4
0021 1 ! DD     (25)  DEPTH                                REAL*4
0022 1 ! NNBT                                INTEGER*2
0023 1 ! T      (25)  TEMPERATURE                           REAL*4
0024 1 ! TT     (25)  TEMPERATURE                           REAL*4
0025 1 ! VEL    (25)  VELOCITY                             REAL*4
0026 1 !
0027 1      INTEGER*2 NNBT
0028 1      REAL*4 D,DD,T,TT,VEL
0029 1
0030 1      COMMON /DTV/ D(25),T(25),VEL(25),DD(25),TT(25),NNBT
0031 1 !-----END DTV-----
0032 1
0033 1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0034 1 ! -----
0035 1 ! DEPDIF DEPTH DIFFERENCE                           REAL*4
0036 1 ! DLYR   BT LAYER DEPTH                             REAL*4
0037 1 ! HLYR   HISTORICAL LAYER DEPTH                     REAL*4
0038 1 ! I      LAYER POSITION FLAG                         INTEGER*2
0039 1 ! NBT    NUMBER OF BT POINTS                         INTEGER*2
0040 1 ! NDLYR  BT LAYER'S POSITION IN ARRAY                 INTEGER*2
0041 1
0042 1 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0043
0044      INTEGER*2 I,NBT,NDLYR
0045      REAL*4 DEPDIF,DLYR,HLYR
0046
0047 1-----MODIFY LAYER DEPTH IF REQUIRE
0048      IF(DLYR.LT.100..AND.DLYR.LE.HLYR) GO TO 25 ! SKIP MODIFICATION
0049      IF(DLYR.GT.HLYR+50.)THEN ! LAYER DEEPER THAN HISTORICAL
0050          DEPDIF=HLYR+50.-DLYR ! SET DEPTH DIFF
0051          DLYR=HLYR+50. ! SET MAX BT LAYER
0052          END IF ! END IF BLOCK
0053      IF(HLYR-50..GE.2500.) THEN ! HISTORICAL LAYER DEEPER THA
0054          NBT=NDLYR ! DO NOT CHANGE LAYER DEPTH
0055          GOTO 25 ! TO 2500 FEET OR DEEPER.
0056          END IF ! END IF BLOCK
0057      IF(DLYR.LT.HLYR-50.)THEN ! LAYER LESS THAN HISTORICAL-
0058          DEPDIF=HLYR-50.-DLYR ! SET DEPTH DIFFERENCE
0059          DLYR=HLYR-50. ! SET MIN BT LAYER

```

```

0060          END IF                                ! END IF BLOCK
0061          IF(D(NDLYR).EQ.DLYR) GO TO 25          ! SKIP DEPTH/VELOCITY CALCULA
0062          D(NDLYR)=DLYR                          ! DEPTH
0063          VEL(NDLYR)=VEL(NDLYR)+DEPDIF/61.        ! ALLOW FOR DEPTH DIFF
0064
0065          !-----ENSURE LAYER DEPTH > PREVIOUS DEPTH IN ARR
0066 25          IF(NDLYR.LE.2) GO TO 100              ! SKIP NEXT
0067          IF(D(NDLYR).GT.D(NDLYR-1)) GOTO 100    ! SKIP NEXT
0068          IF(NDLYR.GT.NBT) NDLYR=NBT              ! CHECK NDLYR
0069          DO 50 I=NDLYR-1,NBT-1                  ! DO FROM LAYER - 1 TO NEXT T
0070          D(I)=D(I+1)                             ! DEPTH
0071          VEL(I)=VEL(I+1)                         ! SS AT LAYER DEPTH
0072 50          CONTINUE                             ! END DO LOOP
0073          NDLYR=NDLYR-1                          ! DECREASE # OF LAYER DEPTHS
0074          NBT=NBT-1                              ! DECREASE # OF BTS
0075          GO TO 25                                ! LOOP BACK
0076
0077          !-----ENSURE LAYER DEPTH < NEXT DEPTH IN A
0078 100         IF(NDLYR.GE.NBT.OR.NDLYR.GE.25) GO TO 155 ! SKIP THIS SECTION
0079          IF(D(NDLYR).LT.D(NDLYR+1)) GOTO 155    ! SKIP THIS SECTION
0080          DO 150 I=NDLYR+1,NBT-1                 ! DO FROM LAYER + 1 TO NEXT T
0081          D(I)=D(I+1)                             ! DEPTH
0082          VEL(I)=VEL(I+1)                         ! SS AT LAYER DEPTH
0083 150         CONTINUE                             ! END DO LOOP
0084          NBT=NBT-1                              ! DECREASE # OF BTS
0085          GO TO 100                              ! LOOP BACK
0086
0087          !-----ENSURE LAYER DEPTH SOUND SPEED IS LOCAL MAXIM
0088 155         IF(NDLYR.LE.2) GO TO 165              ! SKIP THIS SECTION
0089          IF(VEL(NDLYR-1).LT.VEL(NDLYR)) GOTO 165 ! SKIP THIS SECTION
0090          DO 160 I=NDLYR-1,NBT-1                  ! DO FROM LAYER - 1 TO NEXT T
0091          D(I)=D(I+1)                             ! DEPTH
0092          VEL(I)=VEL(I+1)                         ! SS AT LAYER DEPTH
0093 160         CONTINUE                             ! END DO LOOP
0094          NBT=NBT-1                              ! DECREASE # OF BTS
0095          NDLYR=NDLYR-1                          ! DECREASE # OF LAYER DEPTHS
0096          GO TO 155                              ! LOOP BACK
0097
0098 165         IF(NDLYR.GE.NBT.OR.NDLYR.GE.25) GO TO 200 ! SKIP THIS SECTION
0099          IF(VEL(NDLYR+1).LT.VEL(NDLYR)) GOTO 200 ! SKIP THIS SECTION
0100          DO 170 I=NDLYR+1,NBT-1                  ! DO FROM LAYER + 1 TO NEXT T
0101          D(I)=D(I+1)                             ! DEPTH
0102          VEL(I)=VEL(I+1)                         ! SS AT LAYER DEPTH
0103 170         CONTINUE                             ! END DO LOOP
0104          NBT=NBT-1                              ! DECREASE # OF BTS
0105          GO TO 165                              ! LOOP BACK
0106
0107 200         RETURN                                ! RETURN TO CALLING ROUTINE
0108          END                                    ! END SUBROUTINE

```



```

0001      SUBROUTINE MAP(MAPFLG)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: MAP
0005      ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006      ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007      ! DATE: 1982 & 3/84 (FORTRAN 77)
0008      ! FUNCTION: THIS ROUTINE IS THE MAPPING ROUTINE FOR NUSC.
0009      !              DESIGNED TO RETRIEVE AND GRAPHICALLY DISPLAY MGS PROVINCE
0010      !              AREAS, SOUND SPEED INDEX AREAS, AND BOTTOM DEPTH INFOR-
0011      !              TION AS INPUTS TO THE SIMAS PROGRAM.
0012      ! INPUTS: HARD COPY SELECTION, OPERATOR SELECTION TO UPDATE
0013      !              PARAMETERS OR NOT. VARIABLES IN COMMONS.
0014      ! OUTPUTS: CRT PROMPTING MESSAGES TO OPERATOR
0015      ! MODULES CALLED: CRUNCH,ERRSET,ERRTST,FLOOR,FSETUP,GRAPH,HRDCPY,INDX,
0016      !              SETOAC,SETPOS
0017      ! CALLED BY: ENVIRN, FORCST
0018      ! NOTE: MAP PROGRAM REV C
0019      ! =====
0020      ! NOTE:
0021      ! FORMULAS DEFINING MERCATOR PROJECTION ARE:
0022      ! DISTANCE PER DEGREE LAT= ERAD*PI/180*COS(CENTER LATITUDE)
0023      ! DISTANCE PER DEGREE LNG= ERAD*PI/180
0024      ! RATIO OF THE GRAPH'S HEIGHT TO WIDTH=1/COS(CENTER LATITUDE)
0025      ! TO REDUCE EXECUTION TIME, LINEAR FACTORS WILL BE DISREGARDED,
0026      ! SO X IS GRAPHED IN FIFTIETHS OF A DEGREE FROM THE BASE
0027      ! LONGITUDE, AND Y IS GRAPHED AS THE TANGENT OF THE LATITUDE.
0028      ! =====
0029      !
0030      INCLUDE 'MAP.PAR'
0031      1  PARAMETER STOLEN=3800
0032      1  PARAMETER SEGLEN=60, POLLEN=40
0033      1  PARAMETER WRKLEN=1000, NDXLEN=300
0034      1  PARAMETER MAXDTY=3
0035      1  PARAMETER TOL=3
0036      1  PARAMETER DEG=57.2957795
0037      1  PARAMETER RAD=.017453293
0038      1  PARAMETER PI=3.14159265
0039      1  PARAMETER ERAD=3440.3
0040      1  PARAMETER S251=63001
0041      1  PARAMETER TW015=32768
0042      1
0043      1  ! INTEGER*2 MAXDTY,NDXLEN,POLLEN,SEGLEN,STOLEN,TOL,WRKLEN
0044      1  ! INTEGER*4 S251,TW015
0045      1  ! REAL*4 DEG,ERAD,PI,RAD
0046      1  INCLUDE 'CBC1.INC'
0047      1  ! -----CBC1.INC-----
0048      1  ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0049      1  ! -----
0050      1  ! BCOORD (-12:12,2) 7
0051      1  !
0052      1  ! INTEGER*2 BCOORD(-12:12,2)
0053      1
0054      1  ! COMMON /CBC/ BCOORD
0055      1  ! -----END CBC1.INC-----
0056      1
0057      INCLUDE 'CBT.INC'

```

MAP

14-Dec-1984 08:35:15

14-Dec-1984 08:35:14

```

0058 1 ! -----CBT.INC-----
0059 1 ! VARBL  SIZE  PURPOSE                      TYPE  RANGE
0060 1 ! -----
0061 1 ! BTRANS (4,4)  ?
0062 1 !
0063 1          INTEGER*2 BTRANS(4,4)
0064 1
0065 1          COMMON /CBT/  BTRANS
0066 1 ! -----END CBT.INC-----
0067 1
0068 1          INCLUDE 'CFILE.INC'
0069 1 ! -----CFILE.INC-----
0070 1 ! VARBL  SIZE  PURPOSE                      TYPE  RANGE
0071 1 ! -----
0072 1 ! FNAME  (21)  MAP FILE NAME                CHAR
0073 1 ! OPEN      OPEN FLAG                      LOGICAL*1 .FALSE.
0074 1 !
0075 1          LOGICAL*1  OPEN
0076 1          CHARACTER*1 FNAME(21)
0077 1
0078 1          COMMON /CFILE/  OPEN,FNAME
0079 1 ! -----END CFILE.INC-----
0080 1
0081 1          INCLUDE 'CL.INC'
0082 1 ! -----CL.INC-----
0083 1 ! VARBL  SIZE  PURPOSE                      TYPE  RANGE
0084 1 ! -----
0085 1 ! LATMAX      MAXIMUM LATITUDE                INTEGER*2
0086 1 ! LATMIN      MINIMUM LATITUDE                INTEGER*2
0087 1 ! LNGMAX      MAXIMUM LONGITUDE               INTEGER*2
0088 1 ! LNGMIN      MINIMUM LONGITUDE               INTEGER*2
0089 1 !
0090 1          INTEGER*2 LATMIN,LATMAX,LNGMIN,LNGMAX
0091 1
0092 1          COMMON /CL/  LATMIN,LATMAX,LNGMIN,LNGMAX
0093 1 ! -----END CL.INC-----
0094 1
0095 1          INCLUDE 'CLOC.INC'
0096 1 ! -----CLOC.INC-----
0097 1 ! VARBL  SIZE  PURPOSE                      TYPE  RANGE
0098 1 ! -----
0099 1 ! BLAT      BASE LATITUDE                      REAL*4
0100 1 ! BLNG      BASE LONGITUDE                     REAL*4
0101 1 ! LAT       LATITUDE OF SHIP'S LOCATION        REAL*4
0102 1 ! LNG       LONGITUDE OF SHIP'S LOCATION        REAL*4
0103 1 ! NMLT50    # OF NAUTICAL MILES PER 50TH DEGREE REAL*4
0104 1 !           OF LATITUDE
0105 1 ! NMLG50    # OF NAUTICAL MILES PER 50TH DEGREE REAL*4
0106 1 !           OF LONGITUDE
0107 1 !
0108 1          REAL*4  LAT,LNG,BLAT,BLNG,NMLT50,NMLG50
0109 1
0110 1          COMMON /CLOC/  LAT,LNG,BLAT,BLNG,NMLT50,NMLG50
0111 1 ! -----END CLOC.INC-----
0112 1          INCLUDE 'CLOG.INC'
0113 1 ! -----CLOG.INC-----
0114 1 ! VARBL  SIZE  PURPOSE                      TYPE  RANGE

```

```

0115 1 ! -----
0116 1 ! CNVRT(-1:0)
0117 1 ! DG
0118 1 ! DL
0119 1 !
0120 1          BYTE    CNVRT(-1:0),DG,DL
0121 1
0122 1          COMMON /CLOG/  CNVRT,DL,DG
0123 1 ! -----END CLOG.INC-----
0124          INCLUDE 'CNAME.INC'
0125 1 ! -----CNAME.INC-----
0126 1 !  VARBL  SIZE      PURPOSE                                TYPE  RANGE
0127 1 !  -----
0128 1 !  OAC
0129 1 !  ONAME   (25)    OCEAN NAME
0130 1 !  DNAME   ( )    'MGS', 'SSP', OR 'BDEPTH' TEXT STRINGS
0131 1 !
0132 1          INTEGER*4 ONAME(25),DNAME(2*MAXDTY)
0133 1          INTEGER*2 OAC
0134 1
0135 1          COMMON /CNAME/  ONAME,DNAME,OAC
0136 1 ! -----END CNAME.INC-----
0137 1
0138          INCLUDE 'CS.INC'
0139 1 ! -----CS-----
0140 1 !  VARBL  SIZE      PURPOSE                                TYPE  RANGE
0141 1 !  -----
0142 1 !  S      -1,3800    POLYGON AND SEGMENT STORAGE ARRAY      REAL*4
0143 1 !  STOLEN          STORAGE ARRAY LENGTH (FOR SEGS & POLYS) PARM
0144 1 !
0145 1          REAL*4   S(-1:STOLEN)
0146 1
0147 1          COMMON /CS/    S
0148 1 ! -----CS-END-----
0149 1
0150          INCLUDE 'CTSK.INC'
0151 1 ! -----CTSK.INC-----
0152 1 !  VARBL  SIZE      PURPOSE                                TYPE  RANGE
0153 1 !  -----
0154 1 !  TBDPTH
0155 1 !  TFLG
0156 1 !  TLAT   (3)
0157 1 !  TLNG   (3)
0158 1 !  TMGS
0159 1 !  TOAC
0160 1 !  TSSP
0161 1 !
0162 1          REAL*4      TMGS,TSSP,TBDPTH
0163 1          INTEGER*2    TFLG,TOAC,TLAT(3),TLNG(3)
0164 1
0165 1          COMMON /CTSK/  TFLG,TOAC,TLAT,TLNG,TMGS,TSSP,TBDPTH
0166 1 ! -----END CTSK.INC-----
0167 1
0168          INCLUDE 'ENVN.INC'
0169 1 ! -----ENVN-----
0170 1 !  VARBL  SIZE      PURPOSE                                TYPE  RANGE
0171 1 !  -----

```

```

0172 1 ! BIO      (2)      BIOLOGICAL BACK SCATTERING REAL*4      -57. & -47.
0173 1 ! DLYR      LAYER DEPTH REAL*4
0174 1 ! MGS      MGS PROVINCE INTEGER*2
0175 1
0176 1          REAL*4      BIO,DLYR
0177 1          INTEGER*2 MGS
0178 1          DATA BIO/-57.,-47./
0179 1
0180 1          COMMON /ENVN/ BIO(2),DLYR,MGS
0181 1
0182 1 ! -----END ENVN-----
0183          INCLUDE 'MAPLOC.INC'
0184 1 ! -----MAPLOC-----
0185 1 ! VARBL  SIZE  PURPOSE  TYPE  RANGE
0186 1 ! -----
0187 1 ! MPINDX      SSP INDEX      INTEGER*2
0188 1 ! MAPLAT (4)  LATITUDE      INTEGER*2
0189 1 ! LONG (4)   LONGITUDE      INTEGER*2
0190 1 ! NMAREA (20) AREA OCEAN NAME  BYTE
0191 1 ! NOC      NUMBER OF OCEAN  INTEGER*2
0192 1 ! RCZ      RANGE TO CONVERG. ZONE REAL*4
0193 1
0194 1          REAL*4      RCZ
0195 1          INTEGER*2 MPINDX,MAPLAT,LONG,NOC
0196 1          BYTE      NMAREA(20)
0197 1
0198 1          COMMON /LOC/ MAPLAT(4),LONG(4),NOC,MPINDX,RCZ,NMAREA
0199 1
0200 1 ! -----END MAPLOC-----
0201          INCLUDE 'MAPSVP.INC'
0202 1 ! -----MAPSVP-----
0203 1 ! VARBL  SIZE  PURPOSE  TYPE  RANGE
0204 1 ! -----
0205 1 ! BDF      BOTTOM DEPTH IN FATHOMS REAL*4
0206 1 ! BIOP     BIOLOGICAL BACK SCATTERING COEF REAL*4
0207 1 ! BTDATE (9) DATE OF LAST BT INPUT  BYTE
0208 1 ! BTTIME (8) TIME OF LAST BT INPUT  BYTE
0209 1 ! C (50)   VELOCITY (PAIRED WITH Z FOR SVP) REAL*4
0210 1 ! CC (50)  VELOCITY (PAIRED WITH ZZ FOR SVP) REAL*4
0211 1 ! CS      SOUND VELOCITY AT SURFACE REAL*4
0212 1 ! MAPDEG  TEMPERATURE (DEG) REAL*4      57.2957795
0213 1 ! EL      LAYER DEPTH DATA
0214 1 ! F      FREQUENCY REAL*4
0215 1 ! GRDS    GRIDS REAL*4      0.0164
0216 1 ! ITO     MINIMAL 2-WAY TRAVEL TIME INTEGER*2
0217 1 ! MGSOP   MGS PROVINCE NUMBER INTEGER*2
0218 1 ! N      # OF DEPTH/VELOCITY PAIRS INTEGER*2
0219 1 ! NN     # OF DEPTH/VELOCITY PAIRS INTEGER*2
0220 1 ! MAPPI   MATHEMATICAL CONSTANT PI REAL*4      3.1415927
0221 1 ! SNDATE (9) DATE SYS PARMS LAST UPDATED  BYTE
0222 1 ! SNTIME (8) TIME SYS PARMS LAST UPDTAED  BYTE
0223 1 ! SYDATE (9) CURRENT DATE READ FROM SYSTEM  BYTE
0224 1 ! SYTIME (8) CURRENT TIME READ FROM SYSTEM  BYTE
0225 1 ! TMP     TEMPERATURE REAL*4
0226 1 ! UMKZ    BOTTOM BACK SCATTERING COEF. REAL*4      -28.0
0227 1 ! WS      WIND SPEED REAL*4
0228 1 ! Z (50)  DEPTH OF POINT OF SOUND SPEED REAL*4

```

```

0229 1 ! ZZ      (50)      DEPTH OF POINT OF SOUND SPEED      REAL*4
0230 1
0231 1          INTEGER*2 ITO,MGSOP,N,NN
0232 1          REAL*4    BDF,BIOP,C(50),CC(50),CS,MAPDEG,EL,F,GRDS
0233 1          REAL*4    MAPPI,TMP,UMKZ,WS,Z(50),ZZ(50)
0234 1          BYTE      SYDATE(9),SYTIME(8),BTDATE(9),BTTIME(8)
0235 1          BYTE      SNDATE(9),SNTIME(8)
0236 1          DATA     MAPPI,MAPDEG,GRDS/3.1415927,57.2957795,0.0164/
0237 1          DATA     UMKZ/-28./
0238 1
0239 1          COMMON /SVP/ F,N,Z,C,EL,MGSOP,BDF,WS,CS,TMP,BIOP,
0240 1              1      UMKZ,MAPPI,MAPDEG,GRDS,ITO,ZZ,CC,NN,
0241 1              2      SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME
0242 1 ! -----MAPSVP-END-----
0243
0244 !
0245 !  VARBL  SIZE      PURPOSE                                TYPE      RANGE
0246 ! -----
0247 !  DEAST      DISPLAY SHIFT FLAG: EAST                    BYTE
0248 !  DELIM (6)   DATA TYPE DELIMITER                        REAL*4
0249 !  DNORTH     DISPLAY SHIFT FLAG: NORTH                    BYTE
0250 !  DSOUTH     DISPLAY SHIFT FLAG: SOUTH                    BYTE
0251 !  DTYPE      DATA TYPE LOOP COUNTER                      INNTEGER*2
0252 !  DWEST      DISPLAY SHIFT FLAG: WEST                      BYTE
0253 !  ERR        ERROR TEST FLAG                              INTEGER*2
0254 !  FLOOR      FUNCTION                                       INTEGER*2
0255 !  I          LOOP COUNTER                                  INTEGER*2
0256 !  INFO (3)   MGS, SSP, BDEPTH INFO                          REAL*4
0257 !  ILAT       LATITUDE                                       INTFGR*2
0258 !  ILNG       LONGITUDE                                       INTEGER*2
0259 !  INDX       FUNCTION                                       INTEGER*2
0260 !  IX (4)     BASE X COORD OF 4 QUADRANTS                    INTEGER*2
0261 !  IY (4)     BASE Y COORD OF 4 QUADRANTS                    INTEGER*2
0262 !  MAPFLG     MAP FLAG                                       INTEGER*2
0263 !  MEAST      EAST ADJUSTMENT                                BYTE
0264 !  MNORTH     NORTH ADJUSTMENT                              BYTE
0265 !  MSOUTH     SOUTH ADJUSTMENT                              BYTE
0266 !  MWEST      WEST ADJUSTMENT                                BYTE
0267 !  NEW        DATA TYPE DELIMITER                          REAL*4
0268 !  R          PROGRAM FLAG                                    BYTE
0269 !  RECNDX (300) RECORD INDEX                                INTEGER*2
0270 !  SHIFT      LATITUDE OR LONGITUDE SHIFT                    BYTE
0271 !
0272 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0273
0274          REAL*4    DELIM(2*MAXDTY),INFO(MAXDTY),NEW
0275          INTEGER*2  DTYPE,ERR,FLOOR,I,ILAT,ILNG,INDX,IX(4),IY(4)
0276          INTEGER*2  MAPFLG,RECNDX(NDXLEN)
0277          BYTE      DEAST,DNORTH,DSOUTH,DWEST,MEAST,MNORTH
0278          BYTE      MSOUTH,MWEST,R,SHIFT
0279          EQUIVALENCE (INFO(1),TMGS),(INFO(2),TSSP),(INFO(3),TBDPTH)
0280          DATA     DELIM /1.,9.,1.,99.,0.,5500./
0281
0282          DATA     BCOORD /0,5,10,0,5,10,3*5,3*10,0,3*5,3*0,10,5,0,10,
0283 1              5,0,3*10,3*5,10,5,0,10,5,3*0,5,10,0,5,10,3*0,3*5/
0284          DATA     BTRANS /1,2,4,5,7,10,8,11,-2,-3,-5,-6,-8,-11,-9,-12/
0285          DATA     FNAME /11*'M','A','P','@','@','.',',','D','A','T',';',',','O'/'

```

```

0286      DATA OPEN  /.FALSE./
0287      DATA CNVRT  /1,0/
0288      DATA ONAME  /'Nort','h Pa','ciFi','c Oc','ean','Nort','h At',
0289      1          'lant','ic O','cean','Medi','terr','anea','n Se',
0290      2          'a','Indi','an O','cean',' ',' ','Norw','egia',
0291      3          'n Se','a',' ' /
0292      DATA DNAME  /'MGS',0,'SSP',0,'BDEP','TH' /
0293      DATA S      /STOLEN,0,STOLEN*0/
0294      DATA TFLG   /.TRUE./
0295
0296      !-----PRELIMINARIES-----
0297      CALL ERRSET(29,,,FALSE.,,.FALSE.)! NON-EXISTANT FILE
0298      CALL ERRSET(39,,,FALSE.,,.FALSE.)! FILE READ ERROR
0299      CALL ERRSET(63,,,FALSE.,,.FALSE.)! END-OF-FILE ENCOUNTERED
0300      CALL ERRSET(72,,,FALSE.,,.FALSE.)! FLOATING POINT OVERFLOW
0301      C      CALL ERRSET(75,,,FALSE.,,.FALSE.)! INTEGER*2 CONVERSION ERROR
0302      DO 1 I=-12,12          ! SET UP BORDER COORDINATES
0303          BCOORD(I,1)=BCOORD(I,1)*50 ! INITIALIZE BORDER COORDS
0304          BCOORD(I,2)=BCOORD(I,2)*50 ! INITIALIZE BORDER COORDS
0305      1      CONTINUE          ! END DO LOOP
0306      2      CALL SETOAC        ! GET OCEAN AREA
0307      CALL FSETUP(RECNDX,R)    ! SET UP FILE "MAP"(OAC)"A"
0308      IF (R) GOTO 2            ! IF ERROR ASK OAC AGAIN
0309      CALL SETPOS(1,R)         ! GET LATITUDE
0310      IF (R) CALL EXIT         ! IF ERROR, STOP PROGRAM
0311      IF (.NOT. TFLG) GOTO 50  ! GO TO TSK TO PASS TO SIMAS
0312      CALL SETPOS(2,R)         ! GET LONGITUDE
0313      IF (R) CALL EXIT         ! IF ERROR, STOP PROGRAM
0314      IF (.NOT. TFLG) GOTO 50  ! GO TO TSK TO PASS TO SIMAS
0315      ILAT=FLOOR(LAT/5.)      ! SET SHIP POSITION:LOWEST OFFSET
0316      ILNG=FLOOR(LNG/5.)      ! SET SHIP POSITION:LOWEST OFFSET
0317
0318      !-----MAIN PROGRAM LOOP FOR THE 3 DATA T
0319      DO 21 DTYPE=1,MAXDTY    ! CALC, DISPLAY DATA TYPES
0320          DL=(DTYPE.LE.2)     ! SET FLAG FOR POLYGON DATA
0321          DG=.NOT.DL          ! SET FLAG FOR NON-POLYGON DATA
0322          INFO(DTYPE)=-3.     ! INFO FLAG BEYOND LEGAL VALUE
0323          IF (RECNDX(INDX(ILAT,ILNG,DTYPE)).EQ.-TWO15) GOTO 17 ! ANY DATA
0324          ! OFFSETS FOR THE GRAPHICS
0325          BLAT=FLOATI(5*ILAT-5) ! BASE LATITUDE
0326          BLNG=FLOATI(5*ILNG)   ! BASE LONGITUDE
0327
0328      !-----ASSUME SHIP IN QUADRANT 1 (TOP LEF
0329          IY(1)=ILAT          ! SET Y COORDINATE
0330          IY(2)=ILAT          ! SET Y COORDINATE
0331          IY(3)=ILAT-1        ! SET Y COORDINATE
0332          IY(4)=ILAT-1        ! SET Y COORDINATE
0333          IX(1)=ILNG          ! SET X COORDINATE
0334          IX(2)=ILNG+1        ! SET X COORDINATE
0335          IX(3)=ILNG          ! SET X COORDINATE
0336          IX(4)=ILNG+1        ! SET X COORDINATE
0337
0338      !=====SHIFT THE GRAPHICS DISPLAY ACCORDI
0339      !=====TO SHIP LOCATION & THE OCEAN AREA BOUNDAR
0340
0341      !-----WEST AND EAST ADJUSTMENTS-----
0342      MWEST=((LNG-5).GE.LNGMIN) ! WEST ADJUSTMENT

```

```

0343      DWEST=.FALSE.                ! DISPLAY SHIFT FLAG: WEST
0344      IF (MWEST) DWEST=(RECNDX(INDX(ILAT,ILNG-1,DTYPE)).GT.-TWO15)
0345      MEAST=((LNG+5).LT.LNGMAX)      ! EAST ADJUSTMENT
0346      DEAST=.FALSE.                ! DISPLAY SHIFT FLAG: EAST
0347      IF (MEAST) DEAST=(RECNDX(INDX(ILAT,ILNG+1,DTYPE)).GT.-TWO15)
0348      SHIFT=(LNG/5-ILNG.LT..5)      ! LATITUDE SHIFT
0349      IF (DWEST.XOR.DEAST) SHIFT=DWEST ! LONGITUDE SHIFT
0350      IF (.NOT.(DWEST.OR.DEAST) .AND. (MWEST.XOR.MEAST))
0351          *      SHIFT=MWEST          ! LONGITUDE SHIFT
0352      IF (.NOT.SHIFT) GOTO 4          ! NO SHIFT
0353      BLNG=BLNG-5.                  ! SHIFT DISPLAY WEST
0354      DO 3 I=1,4                    ! DO FOR FOUR X COORDINATES
0355          IX(I)=IX(I)-1              ! DECREASE X COORDINATE
0356      3      CONTINUE                ! END DO LOOP
0357
0358      !-----SOUTH AND NORTH ADJUSTMENTS-----
0359      4      MSOUTH=(LAT-5.GE.LATMIN) ! SOUTH ADJUSTMENT
0360      DSOUTH=.FALSE.                ! DISPLAY SHIFT FLAG: SOUTH
0361      IF (MSOUTH) DSOUTH=(RECNDX(INDX(ILAT-1,ILNG,DTYPE)).GT.-TWO15)
0362      MNORTH=(LAT+5.LT.LATMAX)      ! NORTH ADJUSTMENT
0363      DNORTH=.FALSE.                ! DISPLAY SHIFT FLAG: NORTH
0364      IF (MNORTH) DNORTH=(RECNDX(INDX(ILAT+1,ILNG,DTYPE)).GT.-TWO15)
0365      SHIFT=(LAT/5-ILAT.GE..5)      ! LATITUDE SHIFT
0366      IF (DNORTH.XOR.DSOUTH) SHIFT=DNORTH ! LONGITUDE SHIFT
0367      IF (.NOT.(DNORTH.OR.DSOUTH) .AND. (MNORTH.XOR.MSOUTH))
0368          *      SHIFT=MNORTH          ! LONGITUDE SHIFT
0369      IF (.NOT.SHIFT) GOTO 6          ! NO SHIFT, SKIP
0370      BLAT=BLAT+5.                  ! SHIFT DISPLAY NORTH
0371      DO 5 I=1,4                    ! DO FOR FOUR Y COORDINATES
0372          IY(I)=IY(I)+1              ! INCREASE Y COORDINATES
0373      5      CONTINUE                ! END DO LOOP
0374
0375      !-----ADJUST SHIP'S LOCATION-----
0376      6      NMLT50=ERAD*PI/(50.*180.) ! # NAUTICAL MILES/50TH DEG LAT
0377      NMLG50=NMLT50*COS((BLAT+5.)*PI/180.) ! # NAUTICAL MILES/50TH DEG L
0378      LAT=50.*(LAT-BLAT)              ! SET UNITS FOR SHIP LOCATION
0379      LNG=50.*(LNG-BLNG)              ! SHIP LOCATION
0380
0381      !-----PROCESS DATA FOR THE 10 DEGREE SQUARE-----
0382      INFO(DTYPE)=-2.                ! RESET FLAG (DATA NOT PROCSSSED)
0383      WRITE(5,7) DNAME(DTYPE*2-1),DNAME(DTYPE*2) ! ANALYZING DATA MSG
0384      C      CALL ERRST(75,ERR)        ! RESET FLAG FOR INTEGER*2 OVERFLOI
0385      CALL CRUNCH(RECNDX,INFO(DTYPE),DTYPE,IY,IX,R) ! PROCESS DATA
0386      C      CALL ERRST(75,ERR)        ! FLAG FOR INTEGER*2 OVERFLOW
0387
0388      !-----DISPLAY DATA (IF DESIRED)-----
0389      WRITE(5,10001)                  ! BELL PROMPT
0390      IF (DG) WRITE(5,10002)          ! ERASE THE WHOLE SCREEN
0391      IF ((DG) .OR. (R) .OR. (ERR-2)) GOTO 8 ! SKIP GRAPHICS
0392      CALL GRAPH(DTYPE)                ! DISPLAY GRAPHICS
0393      8      WRITE(5,10001)            ! PROMPT BELL
0394      IF (DG) TYPE *, 'BOTTOM DEPTH IS IN FATHOMS' ! IF BOTTOM DEPTH
0395      WRITE(5,9) DNAME(DTYPE*2-1),DNAME(DTYPE*2),INT(INFO(DTYPE))
0396      10      READ(5,10003) NEW         ! NEW CURRENT DATA TYPE
0397      IF (NEW.NE.0.) INFO(DTYPE)=NEW ! NEW DATA TYPE
0398      IF (INFO(DTYPE).GE.DELIM(DTYPE*2-1) ! VALID, SKIP NEXT
0399      ( 9      *      .AND. INFO(DTYPE).LE.DELIM(DTYPE*2)) GOTO 12 ! SKIP NEXT

```

```

0400      WRITE(5,11) DNAME(DTYPE*2-1),DNAME(DTYPE*2), ! PROMPT OPERATOR
0401      *      INT(DELM(DTYPE*2-1)),INT(DELM(DTYPE*2))!   FOR VALID
0402      GOTO 10 ! INVALID, TRY AGAIN
0403      12      IF (.NOT.DG) CALL HRDCPY ! NON-POLYGON DATA FLAG
0404      WRITE(5,10002) ! CLEAR SCREEN
0405      LAT=LAT/50.+BLAT ! LOCATION IN DEGREES
0406      LNG=LNG/50.+BLNG ! LOCATION IN DEGREES
0407      GOTO 21 ! DO NEXT DATA TYPE
0408      17      WRITE(5,18)DNAME(DTYPE*2-1), ! PROMPT FOR NEW VALUE
0409      *      DNAME(DTYPE*2),INT(DELM(DTYPE*2-1)),INT(DELM(DTYPE*2))
0410      19      READ(5,10003) NEW ! NEW VALUE
0411      INFO(DTYPE)=NEW ! STORE NEW VALUE
0412      IF (INFO(DTYPE).GE.DELM(DTYPE*2-1)! VALID
0413      *      .AND. INFO(DTYPE).LE.DELM(DTYPE*2)) GOTO 21
0414      WRITE(5,20) ! INVALID WARNING
0415      GOTO 19 ! TRY AGAIN
0416      21      CONTINUE ! END DO LOOP
0417      NOC=OAC ! OCEAN NUMBER
0418      BDF=INFO(3) ! BOTTOM DEPTH IN FATHOMS
0419      MGS=IIFIX(INFO(1)) ! MGS PROVINCE NUMBER
0420      MGSOP=MGS ! MGS PROVINCE NUMBER
0421      MPINDX=IIFIX(INFO(2)) ! SSP INDEX
0422      DO 25 I=1,3 ! DO 3 TIMES
0423      MAPLAT(I)=IABS(TLAT(I)) ! LATITUDE
0424      LONG(I)=IABS(TLNG(I)) ! LONGITUDE
0425      25      CONTINUE ! END DO LOOP
0426      MAPLAT(4)='N' ! LATITUDE
0427      IF(TLAT(1).LT.0) MAPLAT(4)='S' ! LATITUDE
0428      LONG(4)='E' ! LONGITUDE
0429      IF(TLNG(1).LT.0) LONG(4)='W' ! LONGITUDE
0430      GOTO 999 ! RETURN TO CALLING ROUTINE
0431      50      MAPFLG=.FALSE. ! SET MAP FLAG TO FALSE
0432      999      RETURN ! RETURN TO CALLING ROUTINE
0433
0434      !-----FORMAT STATEMENTS-----
0435      7      FORMAT(/26X,'* ANALYZING ',2A4,' DATA *')
0436      9      FORMAT(X,'THE ',2A4,' VALUE IS ',I5,/
0437      *      X,'ENTER NEW VALUE (RETURN KEEPS CURRENT VALUE) '$)
0438      11     FORMAT(X,'INVALID ',2A4,' VALUE MIN=',I2,' MAX=',I4,/,
0439      *      X,'RE-ENTER VALUE ',I5,/)
0440      18     FORMAT(X,'DATA BASE CONTAINS NO ',2A4,' INFORMATION',
0441      *      /,X,'MIN=',I2,' MAX=',I4,' ENTER VALUE ',I5,/)
0442      20     FORMAT(X,'INVALID VALUE, RE-ENTER ',I5,/)
0443      10001  FORMAT(X,'!BEL') ! 4025 BELL PROMPT
0444      10002  FORMAT(X,'!WOR 0') ! 4025 SCREEN ERASE
0445      10003  FORMAT(F9.0) ! FLOATING DECTMAL. POINT FORMAT
0446      END

```



```

0001      SUBROUTINE METRIC(INSSP,D,T,NBT,Z,C,SLNTY,VS1,IERROR)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: METRIC
0005      ! AUTHOR: S. LAFLEUR & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1983 & 12/83 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE METRIC PRODUCES DEPTH, TEMPERATURE,
0008      !             AND SOUND SPEED IN ENGLISH UNITS FROM INPUT DEPTH,
0009      !             TEMPERATURE, AND/OR SOUND SPEED ENTERED IN ENGLISH
0010      !             OR METRIC UNITS.
0011      ! INPUTS:  PARAMETERS PASSED IN.
0012      ! OUTPUTS: PARAMETERS PASSED OUT.
0013      ! MODULES CALLED: LEROY,VELTMP
0014      ! CALLED BY: BT,XBTERR
0015      !
0016      ! VARBL   SIZE      PURPOSE                                TYPE      RANGE
0017      ! -----
0018      ! C       (1)      VELOCITY                                REAL*4
0019      ! D       (1)      DEPTH                                    REAL*4
0020      ! FACTOR   CONSTANT FACTOR                                REAL*4      .3048006
0021      ! I       COUNTER                                        INTEGER*2
0022      ! IERROR   ERROR IN DATA INPUT FLAG                    INTEGER*2
0023      ! INSSP    TYPE OF SSP SELECTED                          INTEGER*2
0024      ! NBT     NUMBER OF BT POINTS                            INTEGER*2
0025      ! SLNTY   SALINITY                                        REAL*4
0026      ! T       (1)      TEMPERATURE                            REAL*4
0027      ! VS1     VELOCITY                                        REAL*4
0028      ! Z       (1)      DEPTH                                    REAL*4
0029      !
0030      INTEGER*2 I,IERROR,INSSP,NBT
0031      REAL*4    C,D,SLNTY,T,VS1,Z
0032
0033      DIMENSION D(1),T(1),Z(1),C(1)
0034      PARAMETER FACTOR = .3048006
0035
0036      DO 100 I = 1,NBT
0037          IF(T(I).LE.25.) THEN
0038              D(I)=D(I)/FACTOR
0039              T(I)=1.8*T(I)+32.
0040              Z(I)=D(I)
0041              CALL LEROY(D(I),T(I),SLNTY,C(I))
0042              GO TO 100
0043          END IF
0044          IF(T(I).GT.25..AND.T(I).LE.38.) THEN
0045              IF(VS1.GE.4900.) THEN
0046                  D(I)=D(I)/FACTOR
0047                  T(I)=1.8*T(I)+32.
0048              END IF
0049              Z(I)=D(I)
0050              CALL LEROY(D(I),T(I),SLNTY,C(I))
0051              GO TO 100
0052          END IF
0053          IF(T(I).GT.38..AND.T(I).LE.100.) THEN
0054              Z(I)=D(I)
0055              CALL LEROY(D(I),T(I),SLNTY,C(I))
0056              GO TO 100
0057          END IF
0058          IF(T(I).GE.1430..AND.T(I).LE.1600.) THEN
0059              D(I)=D(I)/FACTOR

```

```

0060          Z(I)=D(I)                                ! DEPTH
0061          T(I)=T(I)/FACTOR                          ! TEMPERATURE
0062          C(I)=T(I)                                ! VELOCITY
0063          IF(INSSP.NE.5) CALL VELTMP(D(I),C(I),T(I),SLNTY) ! SSP NOT 5
0064          GO TO 100                                ! NEXT I
0065          END IF                                    ! END ID BLOCK
0066          IF(T(I).GE.4700..AND.T(I).LE.5250.) THEN ! CASE >=4700 TO <=52
0067                                                  ! ENGLISH SOUND SPEED
0068          Z(I)=D(I)                                ! DEPTH
0069          C(I)=T(I)                                ! VELOCITY
0070          IF(INSSP.NE.5) CALL VELTMP(D(I),C(I),T(I),SLNTY) ! SSP NOT 5
0071          GO TO 100                                ! NEXT I
0072          END IF                                    ! END IF BLOCK
0073          WRITE(5,10) I,D(I),T(I)                  ! DATA POINT OUT OF RANG
0074          READ(5,20) IERROR                         ! RE-ENTER PROFILE
0075          IERROR = 1                                ! ERROR FLAG SET
0076          GO TO 999                                ! RETURN TO CALLING ROUT
0077          100   CONTINUE                           ! END DO LOOP
0078          999   RETURN                              ! RETURN TO CALLING ROUT
0079
0080          !-----FORMAT STATEMENT-----
0081          10     FORMAT(' DATA POINT OUT OF RANGE IN 'SUBROUTINE METRIC' '//
0082                1      I5,2F7.2//
0083                2      ' PLEASE RE-ENTER THE PROFILE'/
0084                3      ' OR USE LAST PROFILE AND EDIT THIS POINT'////////
0085                4      ' **** HIT RETURN ****',T45,$)
0086          20     FORMAT(I5)
0087          END                                         ! END SUBROUTINE

```

```

0001      SUBROUTINE MOVE(NEWX,NEWY)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: MOVE
0005      ! AUTHOR: J. CASCIO, W. WACHTER(FORTRAN 77), NUSC/NL, CODE 3333
0006      ! DATE: 1981 & 9/84 (FORTRAN 77)
0007      ! FUNCTION: MOVES A VECTOR FROM THE CURRENT POSITION KBEAMX,KBEAMY
0008      !              TO NEWX,NEWY
0009      ! INPUTS: COORDINATED TO MOVE TO
0010      ! OUTPUTS: MOVED BEAM
0011      ! MODULES CALLED: NONE
0012      ! CALLED BY: AXIS, BOX, GRID, MOVEU, STRING, SVPGRF
0013      !
0014      INCLUDE 'TK4025.INC'
0015 1 !-----TK4025-----
0016 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0017 1 ! -----
0018 1 ! KBEAMX      CURRENT BEAM X POSITION                INTEGER*2
0019 1 ! KBEAMY      CURRENT BEAM Y POSITION                INTEGER*2
0020 1
0021 1      INTEGER*2 KBEAMX,KBEAMY
0022 1
0023 1      COMMON/TK4025/KBEAMX,KBEAMY
0024 1 !-----TK4025 END-----
0025 1
0026 1 ! VARBL  SIZE PURPOSE                                TYPE  RANGE
0027 1 ! -----
0028 1 ! NEWX      X RASTER COORD OF CURRENT BEAM POSITION INTEGER*2
0029 1 ! NEWY      Y RASTER COORD OF CURRENT BEAM POSITION INTEGER*2
0030 1
0031 1      INTEGER*2 NEWX,NEWY
0032 1
0033 1      KBEAMX=NEWX                                ! X COORD OF CURRENT BEAM POSITION
0034 1      KBEAMY=NEWY                                ! Y COORD OF CURRENT BEAM POSITION
0035 1      WRITE(5,1) KBEAMX,KBEAMY                    ! MOVE BEAM
0036 1      RETURN                                        ! RETURN TO CALLING ROUTINE
0037 1
0038 1 !-----FORMAT STATEMENT-----
0039 1      1 FORMAT(' !VEC ',2I5)
0040      END

```

```

0001      SUBROUTINE NOCONV(RRR,LYR)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: NOCONV
0005      ! AUTHOR: G. BROWN & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1974 & 12/83 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE NOCONV IS USED TO OBTAIN A QUICK
0008      !               APPROXIMATION FOR THE RANGE TO THE CONVERGENCE ZONE.
0009      ! INPUTS:   VARIABLES IN COMMONS.
0010      ! OUTPUTS:  PARAMETERS PASSED OUT.
0011      ! MODULES CALLED: ACOUS, INSERT
0012      ! CALLED BY: ENVIRN, FORCST
0013      !
0014      INCLUDE 'SVP.INC'
0015  1 ! -----SVP-----
0016  1 ! VARBL   SIZE      PURPOSE                                TYPE      RANGE
0017  1 ! -----
0018  1 ! BDF      BOTTOM DEPTH IN FATHOMS                        REAL*4
0019  1 ! BIOP     BIOLOGICAL BACK SCATTERING COEF              REAL*4
0020  1 ! BTDATE (9) DATE OF LAST BT INPUT                        BYTE
0021  1 ! BTTIME (8) TIME OF LAST BT INPUT                       BYTE
0022  1 ! C        (50) VELOCITY (PAIRED WITH Z FOR SVP)          REAL*4
0023  1 ! CC       (50) VELOCITY (PAIRED WITH ZZ FOR SVP)        REAL*4
0024  1 ! CS      SOUND VELOCITY AT SURFACE                      REAL*4
0025  1 ! DEG     TEMPERATURE (DEG)                              REAL*4      57.2957795
0026  1 ! EL      LAYER DEPTH                                    DATA
0027  1 ! F       FREQUENCY                                       REAL*4
0028  1 ! GRDS    GRIDS                                           REAL*4      0.0164
0029  1 ! ITO     MINIMAL 2-WAY TRAVEL TIME                       INTEGER*2
0030  1 ! MGSOP   MGS PROVINCE NUMBER                            INTEGER*2
0031  1 ! N       # OF DEPTH/VELOCITY PAIRS                      INTEGER*2
0032  1 ! NN      # OF DEPTH/VELOCITY PAIRS                      INTEGER*2
0033  1 ! PI      MATHEMATICAL CONSTANT PI                      REAL*4      3.1415927
0034  1 ! SNDATE (9) DATE SYS PARMS LAST UPDATED                BYTE
0035  1 ! SNTIME (8) TIME SYS PARMS LAST UPDTAED                 BYTE
0036  1 ! SYDATE (9) CURRENT DATE READ FROM SYSTEM               BYTE
0037  1 ! SYTIME (8) CURRENT TIME READ FROM SYSTEM               BYTE
0038  1 ! TMP     TEMPERATURE                                       REAL*4
0039  1 ! UMKZ    BOTTOM BACK SCATTERING COEF.                   REAL*4      -28.0
0040  1 ! WS      WIND SPEED                                       REAL*4
0041  1 ! Z        (50) DEPTH OF POINT OF SOUND SPEED           REAL*4
0042  1 ! ZZ       (50) DEPTH OF POINT OF SOUND SPEED           REAL*4
0043  1
0044  1      INTEGER*2 ITO,MGSOP,N,NN
0045  1      REAL*4    BDF,BIOP,C(50),CC(50),CS,DEG,EL,F,GRDS
0046  1      REAL*4    PI,TMP,UMKZ,WS,Z(50),ZZ(50)
0047  1      BYTE     SYDATE(9),SYTIME(8),BTDATE(9),BTTIME(8)
0048  1      BYTE     SNDATE(9),SNTIME(8)
0049  1      DATA    PI,DEG,GRDS/3.1415927,57.2957795,0.0164/
0050  1      DATA    UMKZ/-28./
0051  1
0052  1      COMMON /SVP/ F,N,Z,C,EL,MGSOP,BDF,WS,CS,TMP,BIOP,
0053  1      1      UMKZ,PI,DEG,GRDS,ITO,ZZ,CC,NN,
0054  1      2      SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME
0055  1 ! -----SVP-END-----
0056  1
0057  1 ! VARBL   SIZE      PURPOSE                                TYPE      RANGE
0058  1 ! -----
0059  1 ! CV      VERTEX VELOCITY                                REAL*4

```

```

0060      ! DCV                DIFF IN SOUND SPEED                REAL*4
0061      ! GCZ                SPREADING LOSS                    REAL*4
0062      ! HCZ                HORIZONTAL RANGE FOR SOUND SPEED   REAL*4
0063      ! J                  COUNTER                          INTEGER*2
0064      ! LZR                SOUND VELOCITY PROFILE INDEX      INTEGER*2
0065      ! RCZ                COMPUTATION OF RANGE TO CZ        REAL*4
0066      ! RRR                RANGE TO CONVERGENCE ZONE          REAL*4
0067      ! SCZ                SLANT RANGE                        REAL*4
0068      ! TCZ                TRAVEL TIME                       REAL*4
0069      ! TPZ                TIME-VELOCITY GRADIENT            REAL*4
0070      ! VEXCS              VEL EXCESS NEEDED FOR EXISTANCE OF CZ REAL*4
0071      !
0072      ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0073
0074      INTEGER*2 J,LZR
0075      REAL*4    CV,DCV,GCZ,HCZ,RCZ,RRR,SCZ,TCZ,TPZ,VEXCS
0076      DATA VEXCS/20./
0077
0078      RCZ=1000000.
0079      IF (EL.LT.Z(N)) THEN
0080          CALL INSERT(N,Z,C,EL,LZR)
0081          DCV=C(N)-C(LZR)
0082          IF (DCV.GE.VEXCS) THEN
0083              DCV=DCV/9.
0084              CV=C(LZR)-DCV+.01
0085              DO 500 J=1,10
0086                  CV=CV+DCV
0087                  CALL ACOUS(Z,C,N,CV,TCZ,HCZ,GCZ,SCZ,TPZ)
0088                  IF (HCZ.LE.RCZ) THEN
0089                      RCZ=HCZ
0090                  END IF
0091      500      CONTINUE
0092      END IF
0093      ELSE
0094          LZR=N
0095      END IF
0096
0097      RRR=.001*RCZ
0098      RETURN
0099      END

```

! COMPUTATION OF RANGE TO CZ  
! LAYER DEPTH < LAST SVP DEPTH  
! INSERT DEPTH/VEL PAIR  
! SS AT BOTTOM - AT LAYER DEPTH  
! IF SS DIFFERENCE >= 20.0 FT/SEC  
! GET 1/9 THE SOUND SPEED DIFF  
! ADJUSTED SPEED AT LAYER  
! DO TEN TIMES  
! INCREMENT CV BY 1/9 SS DIFF  
! HORIZ RANGE  
! HORIZ RANGE <= RANGE TO CZ  
! RESET RANGE TO CZ  
! END IF BLOCK  
! END DO LOOP  
! END IF BLOCK  
! LAYER DEPTH >= LAST SVP DEPTH  
! SET LAYER DEPTH  
! END IF BLOCK  
! RANGE TO CZ  
! RETURN TO CALLING ROUTINE  
! END SUBROUTINE

```

0001          SUBROUTINE OPNFIL(FTYPE,R)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: OPNFIL
0005      ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006      ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007      ! DATE: 1982 & 6/84 (FORTRAN 77)
0008      ! FUNCTION: SUBROUTINE IS DESIGNED TO OPEN ALL FILES FOR THE "MAP"
0009      !              PROGRAM. ALL FILES ARE DIRECT ACCESS AND HAVE A
0010      !              RECORD SIZE OF 64 WORDS. ONLY FILE NAMES OF THE FORM:
0011      !              "MAP"&(OCEAN AREA NUMBER)&"A" OR "B"
0012      !              WILL BE OPENED. ERRORS IN OPENING FILES WILL BE
0013      !              FLAGGED AND EXECUTION WILL CONTINUE.
0014      ! INPUTS: NAMES NEEDED OF DATA FILE TO BE OPENED
0015      ! OUTPUTS: ERROR FLAG AND OPEN FLAG
0016      ! MODULES CALLED: NONE
0017      ! CALLED BY: CRUNCH, FSETUP
0018      !
0019      INCLUDE 'MAP.PAR'
0020      1      PARAMETER STOLEN=3800
0021      1      PARAMETER SEGLEN=60, POLLEN=40
0022      1      PARAMETER WRKLEN=1000, NDXLEN=300
0023      1      PARAMETER MAXDTY=3
0024      1      PARAMETER TOL=3
0025      1      PARAMETER DEG=57.2957795
0026      1      PARAMETER RAD=.017453293
0027      1      PARAMETER PI=3.14159265
0028      1      PARAMETER ERAD=3440.3
0029      1      PARAMETER S251=63001
0030      1      PARAMETER TWO15=32768
0031      1
0032      1 !      INTEGER*2 MAXDTY,NDXLEN,POLLEN,SEGLEN,STOLEN,TOL,WRKLEN
0033      1 !      INTEGER*4 S251,TWO15
0034      1 !      REAL*4      DEG,ERAD,PI,RAD
0035      INCLUDE 'CFILE.INC'
0036      1 ! -----CFILE.INC-----
0037      1 ! VARBL  SIZE      PURPOSE                                TYPE      RANGE
0038      1 ! -----
0039      1 ! FNAME  (21)      MAP FILE NAME                          CHAR
0040      1 ! OPEN    OPEN FLAG                                LOGICAL*1 .FALSE.
0041      1 !
0042      1      LOGICAL*1 OPEN
0043      1      CHARACTER*1 FNAME(21)
0044      1
0045      1      COMMON /CFILE/ OPEN,FNAME
0046      1 ! -----END CFILE.INC-----
0047      1
0048      INCLUDE 'CNAME.INC'
0049      1 ! -----CNAME.INC-----
0050      1 ! VARBL  SIZE      PURPOSE                                TYPE      RANGE
0051      1 ! -----
0052      1 ! OAC
0053      1 ! ONAME  (25)      OCEAN NAME
0054      1 ! DNAME  ( )      'MGS', 'SSP', OR 'BDEPTH' TEXT STRINGS
0055      1 !
0056      1      INTEGER*4 ONAME(25),DNAME(2*MAXDTY)
0057      1      INTEGER*2 OAC
0058      1
0059      1      COMMON /CNAME/ ONAME,DNAME,OAC

```

```

0060 1 !-----END CNAME.INC-----
0061 1
0062 !
0063 ! VARBL   SIZE      PURPOSE                                TYPE      RANGE
0064 ! -----
0065 ! A              'A'                                CHARACTER*1
0066 ! B              'B'                                CHARACTER*1
0067 ! FTYPE          DATA FILE TYPE                      LOGICAL
0068 ! I              INDEX                                INTEGER*2
0069 ! R              ERROR FLAG                          LOGICAL
0070 !
0071 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0072
0073         INTEGER*2   I
0074         CHARACTER*1 A,B
0075         LOGICAL*1   R,FTYPE
0076         DATA A,B/'A','B'/
0077
0078         R=.FALSE.                ! SET ERROR FLAG
0079         OPEN=.TRUE.              ! SET OPEN FLAG
0080 C         FNAME(5)=B              ! SET FILE NAME TO BE OPENED
0081 C         IF (FTYPE) FNAME(5)=A  ! SET TO INDEX FILE
0082         IF(.NOT.FTYPE) GOTO 200 ! "B: DATA FILES
0083
0084 !-----"A" DATA FILES-----
0085         GOTO(110,120,130,140,150),OAC ! GO TO FILE FOR OEAN REQUEST
0086         GOTO 1                      ! ERROR EXISTS
0087 110      * OPEN (UNIT=4,NAME='MAP1A.DAT;1',ACCESS='DIRECT',RECORDSIZE=32,
0088             READONLY,FORM='UNFORMATTED',TYPE='OLD',ERR=1) ! OPEN N. A
0089             GOTO 3                      ! GO TO RETURN
0090 120      * OPEN (UNIT=4,NAME='MAP2A.DAT;1',ACCESS='DIRECT',RECORDSIZE=32,
0091             READONLY,FORM='UNFORMATTED',TYPE='OLD',ERR=1) ! OPEN N. P
0092             GOTO 3                      ! GO TO RETURN
0093 130      * OPEN (UNIT=4,NAME='MAP3A.DAT;1',ACCESS='DIRECT',RECORDSIZE=32,
0094             READONLY,FORM='UNFORMATTED',TYPE='OLD',ERR=1) ! OPEN MED
0095             GOTO 3                      ! GO TO RETURN
0096 140      * OPEN (UNIT=4,NAME='MAP4A.DAT;1',ACCESS='DIRECT',RECORDSIZE=32,
0097             READONLY,FORM='UNFORMATTED',TYPE='OLD',ERR=1) ! OPEN INDI
0098             GOTO 3                      ! GO TO RETURN
0099 150      * OPEN (UNIT=4,NAME='MAP5A.DAT;1',ACCESS='DIRECT',RECORDSIZE=32,
0100             READONLY,FORM='UNFORMATTED',TYPE='OLD',ERR=1) ! OPEN NORW
0101             GOTO 3                      ! GO TO RETURN
0102
0103 !-----"B" DATA FILES-----
0104 200      GOTO(210,220,230,240,250),OAC ! GO TO FILE FOR OCEAN REQUEST
0105         GOTO 1                      ! ERROR
0106 210      * OPEN (UNIT=4,NAME='MAP1B.DAT;1',ACCESS='DIRECT',RECORDSIZE=32,
0107             READONLY,FORM='UNFORMATTED',TYPE='OLD',ERR=1) ! OPEN N. P
0108             GOTO 3                      ! GO TO RETURN
0109 220      * OPEN (UNIT=4,NAME='MAP2B.DAT;1',ACCESS='DIRECT',RECORDSIZE=32,
0110             READONLY,FORM='UNFORMATTED',TYPE='OLD',ERR=1) ! OPEN N. A
0111             GOTO 3                      ! GO TO RETURN
0112 230      * OPEN (UNIT=4,NAME='MAP3B.DAT;1',ACCESS='DIRECT',RECORDSIZE=32,
0113             READONLY,FORM='UNFORMATTED',TYPE='OLD',ERR=1) ! OPEN MED
0114             GOTO 3                      ! GO TO RETURN
0115 240      * OPEN (UNIT=4,NAME='MAP4B.DAT;1',ACCESS='DIRECT',RECORDSIZE=32,
0116             READONLY,FORM='UNFORMATTED',TYPE='OLD',ERR=1) ! OPEN INDI
0117             GOTO 3                      ! GO TO RETURN
0118 250      OPEN (UNIT=4,NAME='MAP5B.DAT;1',ACCESS='DIRECT',RECORDSIZE=32,

```

```

0119      *      READONLY,FORM='UNFORMATTED',TYPE='OLD',ERR=1) ! OPEN NORW
0120      GOTO 3 ! GO TO RETURN
0121
0122      1      TYPE 2,(FNAME(I),I=1,11) ! ERROR EXISTS IN OPEN FILE
0123      WRITE(5,1001) ! WARN OPERATOR
0124      READ(5,1002) ! PAUSE
0125      R=.TRUE. ! SET ERROR FLAG TO TRUE
0126      OPEN=.FALSE. ! SET OPEN FLAG TO FALSE
0127      3      RETURN ! RETURN TO CALLING PROGRAM
0128
0129      !-----FORMAT STATEMENTS-----
0130      2      FORMAT(' ERROR IN OPENING ''',11A1,''', IN OPNFIL')
0131      1001    FORMAT(23X,'PAUSE (HIT RETURN TO CONTINUE)')$)
0132      1002    FORMAT()
0133      END

```

#### COMMAND QUALIFIERS

```

FORTRAN /CHECK=ALL/LIST/SHOW=(INCLUDE,NOMAP) DBA3:[LAFLEUR]OPNFIL.F77;1

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)
/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD_LINES /NOCROSS_REFERENCE

```

#### COMPILATION STATISTICS

```

Run Time:          1.69 seconds
Elapsed Time:      3.25 seconds
Page Faults:       399
Dynamic Memory:    136 pages

```



```

0001          SUBROUTINE OUTPUT(INPBDF)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: OUTPUT
0005      ! AUTHOR: G. BROWN & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1974 & 11/83 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE OUTPUT ALLOWS OUTPUT A HARDCOPY OF "SIMAS" DATA
0008      ! INPUTS: HARD COPY SELECTION. VARIABLES IN COMMONS.
0009      ! OUTPUTS: HARDCOPY OF SIMAS DATA.
0010      ! MODULES CALLED: NONE
0011      ! CALLED BY: SVPGRF
0012      !
0013          INCLUDE 'DHST.INC'
0014      1 ! -----DHST-----
0015      1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0016      1 ! -----
0017      1 ! SCHNLD                                SOUND CHANNEL LAYER DEPTH  REAL*4
0018      1 !
0019      1 ! REAL*4  SCHNLD
0020      1
0021      1 COMMON /DHST/ SCHNLD
0022      1 ! -----DHST END-----
0023          INCLUDE 'ENVN.INC'
0024      1 ! -----ENVN-----
0025      1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0026      1 ! -----
0027      1 ! BIO      (2)  BIOLOGICAL BACK SCATTERING  REAL*4      -57. & -47.
0028      1 ! DLYR                                LAYER DEPTH          REAL*4
0029      1 ! MGS                                MGS PROVINCE        INTEGER*2
0030      1
0031      1 ! REAL*4  BIO,DLYR
0032      1 ! INTEGER*2 MGS
0033      1 ! DATA BIO/-57.,-47./
0034      1
0035      1 ! COMMON /ENVN/ BIO(2),DLYR,MGS
0036      1
0037      1 ! -----END ENVN-----
0038          INCLUDE 'GRF.INC'
0039      1 ! -----GRF-----
0040      1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0041      1 ! -----
0042      1 ! DBT      (25)  DEPTH OF DEPTH/VEL PAIR                                REAL*4
0043      1 ! IANS                                PREDICTION TYPE          INTEGER*2  -2 TO +2
0044      1 ! ILYR                                INDEX FOR LAYER DEPTH      INTEGER*2
0045      1 ! INBT                                OPERATOR ENTERED # OF BT POINTS  INTEGER*2
0046      1 ! ISVP                                LATEST OR HISTORICAL BT FLAG    INTEGER*2  1 OR 2
0047      1 ! I2000                               SVP INDEX FOR 2000 FT DEPTH      INTEGER*2
0048      1 ! VBT      (25)  VELOCITY FOR DEPTH PAIR  REAL*4          REAL*4
0049      1
0050      1 ! REAL*4  DBT,VBT
0051      1 ! INTEGER*2 IANS,ILYR,INBT,ISVP,I2000
0052      1
0053      1 ! COMMON /GRF/ IANS,ISVP,ILYR,I2000,INBT,DBT(25),VBT(25)
0054      1
0055      1 ! -----END GRF-----
0056          INCLUDE 'LOC.INC'
0057      1 ! -----LOC-----
0058      1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0059      1 ! -----

```

```

0060 1 ! INDX          SSP INDEX          INTEGER*2
0061 1 ! LAT      (4)    LATITUDE          INTEGER*2
0062 1 ! LONG      (4)    LONGITUDE          INTEGER*2
0063 1 ! NMAREA (20)    AREA OCEAN NAME      BYTE
0064 1 ! NOC          NUMBER OF OCEAN        INTEGER*2
0065 1 ! RCZ          RANGE TO CONVERG. ZONE REAL*4
0066 1
0067 1          REAL*4    RCZ
0068 1          INTEGER*2 INDX,LAT, LONG, NOC
0069 1          BYTE      NMAREA(20)
0070 1
0071 1          COMMON /LOC/ LAT(4), LONG(4), NOC, INDX, RCZ, NMAREA
0072 1
0073 1 ! -----END LOC-----
0074          INCLUDE 'OCEANS.INC'
0075 1
0076 1 ! -----OCEANS-----
0077 1 ! VARBL  SIZE  PURPOSE                                     TYPE
0078 1 ! -----
0079 1 ! IOCEAN (50)  ARRAY OF NAMES OF OCEANS                  DATA
0080 1
0081 1          INTEGER*2 IOCEAN
0082 1          DIMENSION IOCEAN(50)
0083 1
0084 1          DATA IOCEAN/'NO','RT','H ','PA','CI','FI','C ','OC','EA','N ',
0085 1          1 'NO','RT','H ','AT','LA','NT','IC','O','CE','AN',
0086 1          2 'ME','DI','TE','RR','AN','EA','N ','SE','A ',
0087 1          3 'IN','DI','AN','O','CE','AN',' ',' ',' ',
0088 1          4 'NO','RW','EG','IA','N ','SE','A ',' ',' '/
0089 1
0090 1          COMMON /OCEANS/ IOCEAN
0091 1
0092 1 ! -----END OCEANS-----
0093          INCLUDE 'SVP.INC'
0094 1 ! -----SVP-----
0095 1 ! VARBL  SIZE  PURPOSE                                     TYPE      RANGE
0096 1 ! -----
0097 1 ! BDF          BOTTOM DEPTH IN FATHOMS                      REAL*4
0098 1 ! BIOP         BIOLOGICAL BACK SCATTERING COEF            REAL*4
0099 1 ! BTDATE (9)    DATE OF LAST BT INPUT                      BYTE
0100 1 ! BTTIME (8)    TIME OF LAST BT INPUT                     BYTE
0101 1 ! C          (50)  VELOCITY (PAIRED WITH Z FOR SVP)        REAL*4
0102 1 ! CC          (50)  VELOCITY (PAIRED WITH ZZ FOR SVP)      REAL*4
0103 1 ! CS          SOUND VELOCITY AT SURFACE                   REAL*4
0104 1 ! DEG         TEMPERATURE (DEG)                          REAL*4      57.2957795
0105 1 ! EL          LAYER DEPTH                                  DATA
0106 1 ! F          FREQUENCY                                      REAL*4
0107 1 ! GRDS        GRIDS                                       REAL*4      0.0164
0108 1 ! ITO         MINIMAL 2-WAY TRAVEL TIME                   INTEGER*2
0109 1 ! MGSOP       MGS PROVINCE NUMBER                         INTEGER*2
0110 1 ! N          # OF DEPTH/VELOCITY PAIRS                    INTEGER*2
0111 1 ! NN         # OF DEPTH/VELOCITY PAIRS                    INTEGER*2
0112 1 ! PI         MATHEMATICAL CONSTANT PI                     REAL*4      3.1415927
0113 1 ! SNDATE (9)    DATE SYS PARMS LAST UPDATED              BYTE
0114 1 ! SNTIME (8)    TIME SYS PARMS LAST UPDTAED              BYTE
0115 1 ! SYDATE (9)    CURRENT DATE READ FROM SYSTEM             BYTE
0116 1 ! SYTIME (8)    CURRENT TIME READ FROM SYSTEM             BYTE
0117 1 ! TMP         TEMPERATURE                                  REAL*4
0118 1 ! UMKZ        BOTTOM BACK SCATTERING COEF.                 REAL*4      -28.0

```

```

0119 1 ! WS          WIND SPEED          REAL*4
0120 1 ! Z          (50) DEPTH OF POINT OF SOUND SPEED REAL*4
0121 1 ! ZZ         (50) DEPTH OF POINT OF SOUND SPEED REAL*4
0122 1
0123 1          INTEGER*2 ITO,MGSOP,N,NN
0124 1          REAL*4    BDF,BIOP,C(50),CC(50),CS,DEG,EL,F,GRDS
0125 1          REAL*4    PI,TMP,UMKZ,WS,Z(50),ZZ(50)
0126 1          BYTE      SYDATE(9),SYTIME(8),BTDATE(9),BTTIME(8)
0127 1          BYTE      SNDATE(9),SNTIME(8)
0128 1          DATA     PI,DEG,GRDS/3.1415927,57.2957795,0.0164/
0129 1          DATA     UMKZ/-28./
0130 1
0131 1          COMMON /SVP/ F,N,Z,C,EL,MGSOP,BDF,WS,CS,TMP,BIOP;
0132 1          1          UMKZ,PI,DEG,GRDS,ITO,ZZ,CC,NN,
0133 1          2          SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME
0134 1 ! -----SVP-END-----
0135          INCLUDE 'SVPl.INC'
0136 1 ! -----SVP1-----
0137 1 ! VARBL  SIZE      PURPOSE                      TYPE      RANGE
0138 1 ! -----
0139 1 ! BUFFER (224)    HISTORICAL DATA FILE BUFFER    REAL*4
0140 1 ! DS          (30) HISTORICAL DEPTH              REAL*4
0141 1 ! J20          # OF DEEP OCEAN DEPTH/VEL PAIRS    INTEGER*2
0142 1 ! NS          TOTAL # OF PAIRS IN HISTORICAL    INTEGER*2
0143 1 ! NSN         MONTH NUMBER (1=JAN.,ETC)          INTEGER*2  1 TO 12
0144 1 ! SLNTY       SALINITY                          REAL*4
0145 1 ! VS          (30) HISTORICAL VELOCITY          REAL*4
0146 1
0147 1          REAL*4    BUFFER,DS,SLNTY,VS
0148 1          INTEGER*2 J20,NSN,NS
0149 1
0150 1          COMMON /SVP1/ J20,BUFFER(224),NSN,SLNTY,DS(30),VS(30),NS
0151 1 ! -----END SVP1-----
0152 1
0153 1 ! VARBL  SIZE      PURPOSE                      TYPE      RANGE
0154 1 ! -----
0155 1 ! I          COUNTER                                INTEGER*2
0156 1 ! IBDF       ROUNDED BOTTOM DEPTH                  INTEGER*2
0157 1 ! IBEG       ARRAY JSPDAT START POINTER            INTEGER*2
0158 1 ! IEND       ARRAY JSPDAT START POINTER            INTEGER*2
0159 1 ! INPBDF     INPUTTED BOTTOM DEPTH IN FATHOMS       INTEGER*2
0160 1 ! IOF        ARRAY IOCEAN STOP POINTER            INTEGER*2
0161 1 ! IOS        ARRAY IOCEAN START POINTER            INTEGER*2
0162 1 ! IWS        ROUNDED WIND SPEED                    INTEGER*2
0163 1 ! JSPDAT (15) 'HISTORICAL LATEST XBT KEYPUNCHED' LABELS DATA
0164 1 ! LAYR        ROUNDED LAYER DEPTH                    INTEGER*2
0165 1 ! M          COUNTER                                INTEGER*2
0166 1 ! RMONTH (12) ARRAY OF NAMES OF MONTHS              DATA
0167 1 ! SCHMXD      SOUND CHANNEL MAX DEPTH                REAL*4
0168 1
0169 1 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMON ***
0170 1
0171 1          INTEGER*2 I,IBDF,IBEG,IEND,INPBDF,IOF,IOS,IWS,JSPDAT,LAYR,M
0172 1          REAL*4    SCHMXD
0173 1          DIMENSION JSPDAT(15)
0174 1          DIMENSION RMONTH(12)
0175 1
0176 1          DATA RMONTH/'JAN ','FEB ','MAR ','APR ',
0177 1          1          'MAY ','JUN ','JUL ','AUG ',

```

```

0178      2      'SEP ','OCT ','NOV ','DEC '/
0179 DATA JSPDAT/'HI','ST','OR','IC','AL','LA','TE','ST',' X','BT',
0180 1      'KE','YP','UN','CH','ED'/
0181
0182 !-----PRELIMINARIES-----
0183      SCHMXD = 1000.      ! SOUND CHANNEL MAX DEPTH
0184      IWS=WS+0.5      ! ROUND WIND SPEED
0185      LAYR=DLYR+0.5      ! ROUND CHANNEL LAYER DEPTH
0186      IBDF=BDF+0.5      ! ROUND BOTTOM DEPTH
0187
0188 !-----WRITE TITLES-----
0189      IF( IANS.NE.2) THEN      ! OPERATIONAL PREDICTION
0190      WRITE(6,11)BTDATE(1),BTDATE(2),      ! WRITE UPDATE DATE & TIME
0191      1      BTTIME(1),BTTIME(2),BTTIME(4),BTTIME(5),BTDATE(4),
0192      2      BTDATE(5),BTDATE(6),BTDATE(8),BTDATE(9)
0193      ELSE      ! FORECAST PREDICTION TYPE
0194      WRITE(6,22) RMONTH(NSN)      ! WRITE FOR MONTH
0195      END IF      ! END IF BLOCK
0196      IOS=(NOC-1)*10+1      ! SET IOCEAN START POINTER
0197      IOF=IOS+9      ! SET IOCEAN STOP POINTER
0198      IF( ISVP.EQ.5) THEN      ! IF(KEYPUNCHED)
0199      WRITE(6,3008) NMAREA,LAT, LONG      ! WRITE OCEAN AREA INFO
0200      ELSE      ! NOT KEYPUNCHED
0201      WRITE(6,3005) (IOCEAN(I),I=IOS,IOF),LAT, LONG! WRITE OCEAN INFO
0202      END IF      ! END IF BLOCK
0203
0204 !-----WRITE DATA FOR OCEAN AREA--
0205      IBEG=(ISVP-1)*5+1      ! SET JSPDAT START POINTER
0206      IF( ISVP.EQ.5) IBEG=11      ! IF(KEYPUNCHED)
0207      IEND=IBEG+4      ! SET JSPDAT STOP POINTER
0208      WRITE(6,3135)(JSPDAT(I),I=IBEG,IEND)! WRITE DATA SOURCE
0209      WRITE(6,3150)      ! WRITE #, DEPTH, VELOCITY
0210      WRITE(6,3160) (I,Z(I),C(I),I=1,N) ! WRITE #, DEPTH, VELOCITY
0211      WRITE(6,3175) SLNTY      ! SALINITY
0212      WRITE(6,3180) MGS      ! MGS AREA
0213      IF( ISVP.EQ.5) THEN      ! KEYPUNCHED
0214      WRITE(6,3162) INPBDF      ! INPUTTED BOTTOM DEPTH
0215      ELSE      ! NOT KEYPUNCHED
0216      WRITE(6,3200) INDX      ! SSP INDEX
0217      WRITE(6,3163) INPBDF      ! INPUTTED BOTTOM DEPTH
0218      WRITE(6,3165) IBDF      ! CORRECTED BOTTOM DEPTH
0219      END IF      ! END IF BLOCK
0220      WRITE(6,3170) IWS,LAYR,RCZ      ! WIND SPEED,LAYER,RANGE CZ
0221      IF(SCHNLD.GT.SCHMXD) WRITE(6,3171)! SOUND CHANNEL INVALID
0222      IF(SCHNLD.LE.SCHMXD) WRITE(6,3172) SCHNLD ! SOUND CHANNEL
0223
0224 !-----FORMAT STATEMENTS-----
0225 11      FORMAT('/' BT DATE AND TIME      ',6A1,'Z ',3A1,' ',2A1)
0226 22      FORMAT(1H1,T22,'DATA TO BE USED FOR FORCASTING:',1A4)
0227 3005      FORMAT(2X,10A2,3X,'LAT:',3(I2,1X),
0228      1      A2,' LONG:',3I3,1X,A2)
0229 3008      FORMAT(2X,20A1,3X,'LAT:',3(I2,1X),A2,' LONG:',3I3,1X,A2)
0230 3135      FORMAT(/T15,'SOUND VELOCITY PROFILE DATA'/24X,5A2)
0231 3150      FORMAT(/T15,'NO.',T25,'DEPTH',T33,'VELOCITY')
0232 3160      FORMAT(T15,I2,1X,2F10.1)
0233 3162      FORMAT(T15,'INPUT BOTTOM DEPTH IS ',I5,' FATHOMS')
0234 3163      FORMAT(T15,'CHART OR FATHOMETER BOTTOM DEPTH IS ',I5,' FATHOMS')
0235 3165      FORMAT(T15,'CORRECTED BOTTOM DEPTH IS ',I5,' FATHOMS')
0236 3170      FORMAT(T15,'TRUE WIND SPEED IS ',I2,' KNOTS'

```

```

0237      1      /T15,'****CANDIDATE ACOUSTIC PATHS****'
0238      2      /T25,'LAYER DEPTH IS ',I6,' FEET'
0239      3      /T25,'CONVERGENCE ZONE RANGE IS ',F5.1,' KYDS')
0240      3171    FORMAT(T25,'SOUND CHANNEL NOT USABLE')
0241      3172    FORMAT(T25,'SOUND CHANNEL AXIS DEPTH IS ',F7.1,' FEET')
0242      3175    FORMAT(/T15,'SALINITY IS ',F6.2)
0243      3180    FORMAT(T15,'MGS AREA IS 'I1)
0244      3200    FORMAT(T15,'SSP AREA IS ',I2)
0245
0246      RETURN
0247      END

```

#### COMMAND QUALIFIERS

```

FORTRAN /CHECK=ALL/LIST/SHOW=(INCLUDE,NOMAP) OUTPUT.F77

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)
/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD_LINES /NOCROSS_REFERENCE

```

#### COMPILATION STATISTICS

```

Run Time:          3.49 seconds
Elapsed Time:      12.77 seconds
Page Faults:       494
Dynamic Memory:    145 pages

```

```

0001      INTEGER*2 FUNCTION PDIST(NDX)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: PDIST
0005      ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006      ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007      ! DATE: 1982 & 6/84 (FORTRAN 77)
0008      ! FUNCTION: THIS SUBROUTINE IS DESIGNED TO ROTATIONAL ANALYSIS UPON A
0009      !               SERIES OF SEGMENTS OF POINTS, RETURNING THE DISTANCE TO
0010      !               THE NEAREST POINT TO THE SHIP.
0011      ! INPUTS: VARIABLES NEEDED FOR ROTATION ANALYSIS
0012      ! OUTPUTS: DISTANCE TO NEAREST POINT TO THE SHIP
0013      ! MODULES CALLED: END1, END2
0014      ! CALLED BY: CRUNCH
0015      !
0016      !=====
0017      ! ASSUMPTIONS MADE BY THIS ALGORITHM:
0018      !      1) LEGIT SEGS: 1-6 VERT BORDER, 7-12 HORZ BORDER, >12 DIGITIZED
0019      !      2) SEGMENTS MUST BE LISTED IN CONNECTED CLOCKWISE ORDER
0020      !      3) NEGATIVE SEGS INDICATE REVERSE ORDER OF POINTS
0021      !      4) ONLY SEGS LISTED ARE THOSE ON OUTER PERIMETER
0022      !=====
0023      !
0024      INCLUDE 'MAP.PAR'
0025      1      PARAMETER STOLEN=3800
0026      1      PARAMETER SEGLEN=60, POLLEN=40
0027      1      PARAMETER WRKLEN=1000, NDXLEN=300
0028      1      PARAMETER MAXDTY=3
0029      1      PARAMETER TOL=3
0030      1      PARAMETER DEG=57.2957795
0031      1      PARAMETER RAD=.017453293
0032      1      PARAMETER PI=3.14159265
0033      1      PARAMETER ERAD=3440.3
0034      1      PARAMETER S251=63001
0035      1      PARAMETER TWO15=32768
0036      1
0037      1 !      INTEGER*2 MAXDTY,NDXLEN,POLLEN,SEGLEN,STOLEN,TOL,WRKLEN
0038      1 !      INTEGER*4 S251,TWO15
0039      1 !      REAL*4      DEG,ERAD,PI,RAD
0040      INCLUDE 'CBC2.INC'
0041      1 ! -----CBC2.INC-----
0042      1 ! VARBL   SIZE      PURPOSE                                TYPE   RANGE
0043      1 ! -----
0044      1 ! BCOORD (25,2)
0045      1 !
0046      1      INTEGER*2 BCOORD(25,2)
0047      1
0048      1      COMMON /CBC/      BCOORD
0049      1 ! -----END CBC2.INC-----
0050      1
0051      INCLUDE 'CLOC.INC'
0052      1 ! -----CLOC.INC-----
0053      1 ! VARBL   SIZE      PURPOSE                                TYPE   RANGE
0054      1 ! -----
0055      1 ! BLAT      BASE LATITUDE                                REAL*4
0056      1 ! BLNG      BASE LONGITUDE                                REAL*4
0057      1 ! LAT      LATITUDE OF SHIP'S LOCATION                    REAL*4
0058      1 ! LNG      LONGITUDE OF SHIP'S LOCATION                    REAL*4
0059      1 ! NMLT50    # OF NAUTICAL MILES PER 50TH DEGREE            REAL*4

```

```

0060 1 !           OF LATITUDE
0061 1 ! NMLG50      # OF NAUTICAL MILES PER 50TH DEGREE      REAL*4
0062 1 !           OF LONGITUDE
0063 1 !
0064 1           REAL*4  LAT,LNG,BLAT,BLNG,NMLT50,NMLG50
0065 1
0066 1           COMMON /CLOC/  LAT,LNG,BLAT,BLNG,NMLT50,NMLG50
0067 1 ! -----END CLOC.INC-----
0068           INCLUDE 'CS.INC'
0069 1 ! -----CS-----
0070 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0071 1 ! -----
0072 1 ! S      -1,3800  POLYGON AND SEGMENT STORAGE ARRAY      REAL*4
0073 1 ! STOLEN          STORAGE ARRAY LENGTH (FOR SEGS & POLYS)  PARM
0074 1 !
0075 1           REAL*4  S(-1:STOLEN)
0076 1
0077 1           COMMON /CS/  S
0078 1 ! -----CS-END-----
0079 1
0080 !
0081 ! VARBL  SIZE PURPOSE                                TYPE  RANGE
0082 ! -----
0083 ! D              MAX DISTANCE BETWEEN 2 POINTS ON EARTH      REAL*4
0084 ! I              LOOP COUNTER                                INTEGER*2
0085 ! J              SEGMENT                                    INTEGER*2
0086 ! N              NUMBER OF SEGMENTS IN A POLYGON              REAL*4
0087 ! NDX           POINTER TO THE # OF SEGMENTS USED            INTEGER*2
0088 ! Q              FACTOR                                      REAL*4
0089 ! R              ROTATIONAL ANGLE                            REAL*4
0090 ! T              FACTOR                                      REAL*4
0091 ! TEMP          FACTOR                                      REAL*4
0092 ! X1            STARTING X COORDINATE                        REAL*4
0093 ! X2            ENDING X COORDINATE                          REAL*4
0094 ! X3            DISTANCE BETWEEN X COORDINATES              REAL*4
0095 ! Y1            STARTING Y COORDINATE                        REAL*4
0096 ! Y2            ENDING Y COORDINATE                          REAL*4
0097 ! Y3            DISTANCE BETWEEN Y COORDINATES              REAL*4
0098 !
0099 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0100
0101           INTEGER*2 I,J,END1,END2,NDX
0102           REAL*4 D,N,Q,R,T,TEMP,X1,Y1,X2,Y2,X3,Y3
0103
0104 ! -----PRELIMINARIES-----
0105           N=10.**5           ! # OF SEGMENTS IN A POLYGON
0106           R=0.              ! RESET ROTATION (DEGREES)
0107           D=(ERAD*PI)**2    ! MAX DIST OF 2 PTS ON EARTH
0108           DO 4 I=1,IIFIX(S(NDX+1))  ! DO FOR ALL SEGMENTS IN POLYGON
0109               J=IIFIX(S(NDX+I+2))  ! SEGMENT
0110
0111 ! -----VERTICAL BORDER SEGMENTS-----
0112           IF (ABS(J).LE.6) THEN  ! VERTICAL BORDER
0113               Y1=FLOATI(END1(NDX,I))  ! STARTING Y COORDINATE
0114               Y2=FLOATI(END2(NDX,I))  ! ENDING Y COORDINATE
0115               X1=FLOATI(BCOORD(J+13,2))  ! STARTING X COORDINATE
0116               X2=X1                  ! ENDING X COORDINATE
0117           END IF                ! END IF BLOCK
0118

```

```

0119 !-----HORIZONTAL BORDER SEGMENTS-----
0120 IF (ABS(J).GT.6.AND.ABS(J).LE.12) THEN ! HORIZONTAL BORDER
0121 X1=FLOATI(END1(NDX,I)) ! STARTING X COORDINATE
0122 X2=FLOATI(END2(NDX,I)) ! ENDING X COORDINATE
0123 Y1=FLOATI(BCOORD(J+13,1)) ! STARTING Y COORDINATE
0124 Y2=Y1 ! ENDING Y COORDINATE
0125 END IF ! END IF BLOCK
0126
0127 !-----DIGITIZED SEGMENTS-----
0128 IF (ABS(J).GT.12) THEN ! DIGITIZED SEGMENT
0129 D=AMIN1(D,S(ABS(J)+2)) ! DIST TO NEAREST PT IN POLYGO
0130 IF (D.LT..25) GOTO 6 ! GO SET DISTANCE TO ZERO
0131 R=R+FLOATI(ISIGN(1,J))*S(11ABS(J)+1) ! ROTATIONAL ANGLE
0132 GOTO 4 ! GO SET DISTANCE
0133 END IF ! END IF BLOCK
0134
0135 !-----ROTATIONAL ANALYSIS-----
0136 X1=(X1-LNG)*NMLG50 ! STARTING X COORDINATE
0137 Y1=(Y1-LAT)*NMLT50 ! STARTING Y COORDINATE
0138 X2=(X2-LNG)*NMLG50 ! ENDING X COORDINATE
0139 Y2=(Y2-LAT)*NMLT50 ! ENDING Y COORDINATE
0140 X3=X2-X1 ! DISTANCE BETWEEN X COORDINATES
0141 Y3=Y2-Y1 ! DISTANCE BETWEEN Y COORDINATES
0142 IF (X3.EQ.0. .AND. Y3.EQ.0.) GOTO 4 ! ZERO DISTANCE
0143 Q=-(X1*X3+Y1*Y3)/(X3**2+Y3**2) ! FACTOR
0144 IF (Q.LE.0.) T=X1**2+Y1**2 ! FACTOR
0145 IF (Q.GT.0. .AND. Q.LT.1.) T=(X1+Q*X3)**2+(Y1+Q*Y3)**2 ! FACT
0146 IF (Q.GE.1.) T=X2**2+Y2**2 ! FACTOR
0147 D=AMIN1(D,T) ! DISTANCE TO NEAREST PT IN PO
0148 T=SQRT((X1**2+Y1**2)*(X2**2+Y2**2)) ! FACTOR
0149 IF (AINT(T*N)/N.EQ.0.) GOTO 6 ! ZERO DISTANCE
0150 TEMP=DEG*ACOS(AINT(((X1*X2+Y1*Y2)/T)*N)/N) ! FACTOR
0151 IF (X2*Y1.NE.Y2*X1) R=R+SIGN(TEMP,(X2*Y1-Y2*X1)) ! SUM SEGMENT
0152 4 CONTINUE ! END DO LOOP
0153 IF (ABS(R).LT.355.) THEN ! SEGMENTS DO NOT ENCIRCLE SHIP
0154 PDIST=ININT(SQRT(D)) ! DISTANCE TO THE NEAREST POINT
0155 ELSE ! SEGMENTS DO FULLY ENCIRCLE SHIP
0156 PDIST=-ININT(SQRT(D)) ! SIGNAL COMPLETE POLYGON
0157 END IF ! END IF BLOCK
0158 GO TO 999 ! RETURN TO CALLING ROUTINE
0159 6 PDIST=0 ! SIGNAL TO CLOSE TO CALL
0160 999 RETURN ! RETURN TO CALLING ROUTINE
0161 END ! END SUBROUTINE

```



```

0001          SUBROUTINE SETOAC
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: SETOAC
0005      ! AUTHOR: E. PETRIDES & P. FRAGEORGIA OF SYSCON, INC
0006      ! RECODED: W. WACHTER, CODE 3333, NUSC/NLL
0007      ! DATE: 1982 & 6/84 (FORTRAN 77)
0008      ! FUNCTION: THIS SUBROUTINE IS DESIGNED TO DETERMINE WHICH DATA
0009      !              BASE FILE THE PROGRAM IS TO USE ACCORDING TO ITS
0010      !              OCEAN AREA CODE.
0011      ! INPUTS: OPERATOR ENTERS DESIRED OCEAN NAME
0012      ! OUTPUTS: DATA BASE FILE TO BE USED
0013      ! MODULES CALLED: NONE
0014      ! CALLED BY: MAP
0015      !
0016          INCLUDE 'MAP.PAR'
0017      1      PARAMETER STOLEN=3800
0018      1      PARAMETER SEGLEN=60, POLLEN=40
0019      1      PARAMETER WRKLEN=1000, NDXLEN=300
0020      1      PARAMETER MAXDTY=3
0021      1      PARAMETER TOL=3
0022      1      PARAMETER DEG=57.2957795
0023      1      PARAMETER RAD=.017453293
0024      1      PARAMETER PI=3.14159265
0025      1      PARAMETER ERAD=3440.3
0026      1      PARAMETER S251=63001
0027      1      PARAMETER TWO15=32768
0028      1
0029      1 !      INTEGER*2 MAXDTY,NDXLEN,POLLEN,SEGLEN,STOLEN,TOL,WRKLEN
0030      1 !      INTEGER*4 S251,TWO15
0031      1 !      REAL*4      DEG,ERAD,PI,RAD
0032      1      INCLUDE 'CFILE.INC'
0033      1 ! -----CFILE.INC-----
0034      1 !  VARBL   SIZE      PURPOSE                                TYPE      RANGE
0035      1 !  -----
0036      1 !  FNAME   (21)      MAP FILE NAME                          CHAR
0037      1 !  OPEN    OPEN     OPEN FLAG                               LOGICAL*1 .FALSE.
0038      1 !
0039      1      LOGICAL*1 OPEN
0040      1      CHARACTER*1 FNAME(21)
0041      1
0042      1      COMMON /CFILE/ OPEN,FNAME
0043      1 ! -----END CFILE.INC-----
0044      1
0045          INCLUDE 'CNAME.INC'
0046      1 ! -----CNAME.INC-----
0047      1 !  VARBL   SIZE      PURPOSE                                TYPE      RANGE
0048      1 !  -----
0049      1 !  OAC
0050      1 !  ONAME   (25)      OCEAN NAME
0051      1 !  DNAME    ()        'MGS', 'SSP', OR 'BDEPTH' TEXT STRINGS
0052      1 !
0053      1      INTEGER*4 ONAME(25),DNAME(2*MAXDTY)
0054      1      INTEGER*2 OAC
0055      1
0056      1      COMMON /CNAME/ ONAME,DNAME,OAC
0057      1 ! -----END CNAME.INC-----
0058      1
0059          INCLUDE 'CTSK.INC'

```

```

0060 1 ! -----CTSK. INC-----
0061 1 ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0062 1 ! -----
0063 1 ! TBDPTH
0064 1 ! TFLG
0065 1 ! TLAT      (3)
0066 1 ! TLNG      (3)
0067 1 ! TMGS
0068 1 ! TOAC
0069 1 ! TSSP
0070 1 !
0071 1          REAL*4      TMGS,TSSP,TBDPTH
0072 1          INTEGER*2    TFLG,TOAC,TLAT(3),TLNG(3)
0073 1
0074 1          COMMON /CTSK/  TFLG,TOAC,TLAT,TLNG,TMGS,TSSP,TBDPTH
0075 1 ! -----END CTSK. INC-----
0076 1
0077 !
0078 ! VARBL  SIZE      PURPOSE                                TYPE  RANGE
0079 ! -----
0080 ! I          LOOP COUNTER                                INTEGER*2
0081 ! I1         ONAME POINTER END                          INTEGER*2 1,6,11,16,21
0082 ! I2         ONAME POINTER END                          INTEGER*2 5,10,15,20,25
0083 ! J          COUNTER                                    INTEGER*2
0084 !
0085 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0086
0087          INTEGER*2 I,I1,I2,J
0088
0089          TYPE *, '!WOR 0'                                ! 4025/27 ERASE COMMAND
0090 1          WRITE(5,2)                                    ! VALID OCEAN AREAS TITLE
0091          DO 4 I=1,5                                       ! DO FOR 5 OCEANS
0092              I1=5*I-4                                     ! POINTER START 1,6,11,16,21
0093              I2=5*I                                       ! POINTER END 5,10,15,20,25
0094          WRITE(5,3) I,(ONAME(J),J=I1,I2) ! VALID OCEAN AREA MENU
0095 4          CONTINUE                                       ! END LOOP
0096          WRITE(5,5)                                       ! OCEAN AREA PROMPT
0097          READ(5,*) OAC                                     ! OCEAN AREA RESPONSE
0098          IF (OAC.LT.1 .OR. OAC.GT.5) GOTO 4 ! INVALID RESPONSE
0099          TOAC=OAC                                         ! SET PARAM TO BE PASSED
0100          FNAME(4)=CHAR(OAC+48)                           ! CHANGE TO ASCII
0101          RETURN                                           ! RETURN TO CALLING ROUTINE
0102
0103 ! -----FORMAT STATEMENT-----
0104 2          FORMAT(4(/),T30,'VALID OCEAN AREAS ARE:')
0105 3          FORMAT(T42,I1,') ',5A4)
0106 5          FORMAT(/X,'ENTER THE CODE OF THE OCEAN AREA DESIRED ',5)
0107          END

```

```

0001      PROGRAM SIMAS
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: SIMAS
0005      ! AUTHOR: G.BROWN, S. LEFLEUR, W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1974, 7/83, & 10/83 (FORTRAN 77)
0007      ! FUNCTION: EXECUTIVE PROGRAM FOR SONAR IN-SITU MODE ASSESSMENT
0008      !                SYSTEM (SIMAS).
0009      !                PROMPTS FOR PREDICTION TYPE AND MENU SELECTION.
0010      !                CONTROLS AND EXECUTES SUBROUTINES NEEDED FOR OPTIONS
0011      !                SELECTED BY OPERATOR.
0012      ! INPUTS:  OPERATOR SELECTION FOR PREDICTION TYPE AND MENU CHOICE.
0013      ! OUTPUTS: NONE
0014      ! MODULES CALLED: ACTVLP, ACTV26, ENVIRN, ERRSET, FORCST, ICLR,
0015      !                OTHERS, PBB, PSSV, RAXIN, SVPTRC, VDS
0016      !
0017      INCLUDE 'ENVN.INC'
0018  1 !-----ENVN-----
0019  1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0020  1 ! -----
0021  1 ! BIO    (2)    BIOLOGICAL BACK SCATTERING REAL*4    -57. & -47.
0022  1 ! DLYR                    LAYER DEPTH              REAL*4
0023  1 ! MGS                    MGS PROVINCE                INTEGER*2
0024  1
0025  1      REAL*4    BIO,DLYR
0026  1      INTEGER*2 MGS
0027  1      DATA BIO/-57.,-47./
0028  1
0029  1      COMMON /ENVN/ BIO(2),DLYR,MGS
0030  1
0031  1 !-----END ENVN-----
0032      INCLUDE 'GRF.INC'
0033  1 !-----GRF-----
0034  1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0035  1 ! -----
0036  1 ! DBT    (25)    DEPTH OF DEPTH/VEL PAIR              REAL*4
0037  1 ! IANS                    PREDICTION TYPE              INTEGER*2    -2 TO +2
0038  1 ! ILYR                    INDEX FOR LAYER DEPTH          INTEGER*2
0039  1 ! INBT                    OPERATOR ENTERED # OF BT POINTS  INTEGER*2
0040  1 ! ISVP                    LATEST OR HISTORICAL BT FLAG      INTEGER*2    1 OR 2
0041  1 ! I2000                   SVP INDEX FOR 2000 FT DEPTH        INTEGER*2
0042  1 ! VBT    (25)    VELOCITY FOR DEPTH PAIR REAL*4          REAL*4
0043  1
0044  1      REAL*4    DBT,VBT
0045  1      INTEGER*2 IANS,ILYR,INBT,ISVP,I2000
0046  1
0047  1      COMMON /GRF/ IANS,ISVP,ILYR,I2000,INBT,DBT(25),VBT(25)
0048  1
0049  1 !-----END GRF-----
0050      INCLUDE 'LOC.INC'
0051  1 !-----LOC-----
0052  1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0053  1 ! -----
0054  1 ! INDX                    SSP INDEX                      INTEGER*2
0055  1 ! LAT    (4)    LATITUDE                                INTEGER*2
0056  1 ! LONG   (4)    LONGITUDE                                INTEGER*2
0057  1 ! NMAREA (20)   AREA OCEAN NAME                          BYTE
0058  1 ! NOC                    NUMBER OF OCEAN                INTEGER*2
0059  1 ! RCZ                    RANGE TO CONVERG. ZONE REAL*4

```

```

0060 1
0061 1      REAL*4      RCZ
0062 1      INTEGER*2  INDX,LAT, LONG, NOC
0063 1      BYTE      NMAREA(20)
0064 1
0065 1      COMMON /LOC/ LAT(4), LONG(4), NOC, INDX, RCZ, NMAREA
0066 1
0067 1 ! -----END LOC-----
0068      INCLUDE 'RXIO.INC'
0069 1 ! -----RXIO-----
0070 1 !  VARBL   SIZE      PURPOSE                                TYPE      RANGE
0071 1 !  -----
0072 1 !  DELTAR      RANGE INCREMENT                                REAL*4
0073 1 !  DEMU        DEPRESSION ANGLE (RECIEVER)                    REAL*4
0074 1 !  DENU        DEPRESSION ANGLE (SONAR)                        REAL*4
0075 1 !  IDL         BEAM DEVIATION LOSS FLAG                        INTEGER*2
0076 1 !  JDL         BEAM DEVIATION LOSS FLAG                        INTEGER*2
0077 1 !  LAMDA       CYCLE RANGE INDEX                              INTEGER*2
0078 1 !  LAMDAB      CYCLE RANGE INDEX - FAST BB                     INTEGER*2
0079 1 !  LAMDAC      NUMBER OF FULL CYCLES                          INTEGER*2
0080 1 !  MU          SOUND SPEED AT TARGET DEPTH                    INTEGER*2
0081 1 !  NPRNT      PLOT MODE FLAG                                  INTEGER*2
0082 1 !  NR         # OF RANGES IN PROP LOSS TABLE                INTEGER*2
0083 1 !  NU          SOUND SPEED AT SONAR DEPTH                      INTEGER*2
0084 1 !  PLRMS      (200)  PROP LOSSES FOR RANGE POINTS             REAL*4
0085 1 !  R           RANGE TO START PROP LOSS                        REAL*4
0086 1 !  RANGE      (200)  RANGE POINTS                              REAL*4
0087 1 !  RMAX        RANGE TO STOP PROP LOSS                         REAL*4
0088 1 !  VBMU        VERTICAL BEAMWIDTH (RECEIVER)                  REAL*4
0089 1 !  VBNU        VERTICAL BEAMWIDTH (SONAR)                     REAL*4
0090 1
0091 1      INTEGER*2  IDL, JDL, LAMDA, LAMDAB, LAMDAC, MU, NPRNT, NR, NU
0092 1      REAL*4     DELTAR, DEMU, DENU, PLRMS, R, RANGE, RMAX, VBMU, VBNU
0093 1
0094 1      DIMENSION PLRMS(200), RANGE(200)
0095 1
0096 1      COMMON /RXIO/ NU, MU, LAMDA, LAMDAC, LAMDAB, NPRNT, R, RMAX, DELTAR,
0097 1      1          IDL, DENU, VBNU, JDL, DEMU, VBMU, NR, RANGE, PLRMS
0098 1
0099 1 ! -----RXIO-END-----
0100 1
0101      INCLUDE 'SVP.INC'
0102 1 ! -----SVP-----
0103 1 !  VARBL   SIZE      PURPOSE                                TYPE      RANGE
0104 1 !  -----
0105 1 !  BDF        BOTTOM DEPTH IN FATHOMS                            REAL*4
0106 1 !  BIOP        BIOLOGICAL BACK SCATTERING COEF                  REAL*4
0107 1 !  BTDATE    (9)      DATE OF LAST BT INPUT                    BYTE
0108 1 !  BTTIME    (8)      TIME OF LAST BT INPUT                     BYTE
0109 1 !  C         (50)     VELOCITY (PAIRED WITH Z FOR SVP)           REAL*4
0110 1 !  CC        (50)     VELOCITY (PAIRED WITH ZZ FOR SVP)         REAL*4
0111 1 !  CS         SOUND VELOCITY AT SURFACE                          REAL*4
0112 1 !  DEG        TEMPERATURE (DEG)                                REAL*4      57.2957795
0113 1 !  EL         LAYER DEPTH                                         DATA
0114 1 !  F          FREQUENCY                                            REAL*4
0115 1 !  GRDS        GRIDS                                              REAL*4      0.0164
0116 1 !  ITO        MINIMAL 2-WAY TRAVEL TIME                          INTEGER*2
0117 1 !  MGSOP      MGS PROVINCE NUMBER                                INTEGER*2
0118 1 !  N          # OF DEPTH/VELOCITY PAIRS                         INTEGER*2

```

```

0119 1 ! NN # OF DEPTH/VELOCITY PAIRS INTEGER*2
0120 1 ! PI MATHEMATICAL CONSTANT PI REAL*4 3.1415927
0121 1 ! SNDATE (9) DATE SYS PARMS LAST UPDATED BYTE
0122 1 ! SNTIME (8) TIME SYS PARMS LAST UPDTAED BYTE
0123 1 ! SYDATE (9) CURRENT DATE READ FROM SYSTEM BYTE
0124 1 ! SYTIME (8) CURRENT TIME READ FROM SYSTEM BYTE
0125 1 ! TMP TEMPERATURE REAL*4
0126 1 ! UMKZ BOTTOM BACK SCATTERING COEF. REAL*4 -28.0
0127 1 ! WS WIND SPEED REAL*4
0128 1 ! Z (50) DEPTH OF POINT OF SOUND SPEED REAL*4
0129 1 ! ZZ (50) DEPTH OF POINT OF SOUND SPEED REAL*4
0130 1
0131 1 INTEGER*2 ITO,MGSOP,N,NN
0132 1 REAL*4 BDF,BIOP,C(50),CC(50),CS,DEG,EL,F,GRDS
0133 1 REAL*4 PI,TMP,UMKZ,WS,Z(50),ZZ(50)
0134 1 BYTE SYDATE(9),SYTIME(8),BTDATE(9),BTTIME(8)
0135 1 BYTE SNDATE(9),SNTIME(8)
0136 1 DATA PI,DEG,GRDS/3.1415927,57.2957795,0.0164/
0137 1 DATA UMKZ/-28./
0138 1
0139 1 COMMON /SVP/ F,N,Z,C,EL,MGSOP,BDF,WS,CS,TMP,BIOP,
0140 1 1 UMKZ,PI,DEG,GRDS,ITO,ZZ,CC,NN,
0141 1 2 SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME
0142 1 !-----SVP-END-----
0143 INCLUDE 'SVPl.INC'
0144 1 !-----SVPl-----
0145 1 ! VARBL SIZE PURPOSE TYPE RANGE
0146 1 ! -----
0147 1 ! BUFFER (224) HISTORICAL DATA FILE BUFFER REAL*4
0148 1 ! DS (30) HISTORICAL DEPTH REAL*4
0149 1 ! J20 # OF DEEP OCEAN DEPTH/VEL PAIRS INTEGER*2
0150 1 ! NS TOTAL # OF PAIRS IN HISTORICAL INTEGER*2
0151 1 ! NSN MONTH NUMBER (1=JAN.,ETC) INTEGER*2 1 TO 12
0152 1 ! SLNTY SALINITY REAL*4
0153 1 ! VS (30) HISTORICAL VELOCITY REAL*4
0154 1
0155 1 REAL*4 BUFFER,DS,SLNTY,VS
0156 1 INTEGER*2 J20,NSN,NS
0157 1
0158 1 COMMON /SVPl/ J20,BUFFER(224),NSN,SLNTY,DS(30),VS(30),NS
0159 1 !-----END SVPl-----
0160 !
0161 ! VARBL SIZE PURPOSE TYPE RANGE
0162 ! -----
0163 ! ICHC MENU SELECTION INTEGER*2 +-1 TO +-11
0164 ! IFL FOR DEFINE FILES INTEGER*2
0165 ! IPRINT PRINT FLAG INTEGER*2 +-1
0166 !
0167 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0168
0169 INTEGER*2 ICHC,IFL,IPRINT
0170
0171 !-----PRELIMINARIES-----
0172 ! RESET FORMAT FIELD LENGTH F
0173 CALL ERRSET(63,.TRUE.,.FALSE.,.FALSE.,.FALSE.,15) ! ERROR MESSAG
0174 40 CALL ICLR ! CLEAR SCREEN
0175
0176 !-----PREDICTION TYPE SELECTION-----
0177 WRITE(5,900) ! PROMPT FOR PREDICTION TYPE

```

```

0178      READ(5,1200) IANS                                ! OPERATIONAL OR FORECAST
0179      IF(IANS.LT.-2.OR.IANS.GT.2.OR.IANS.EQ.0) GO TO 40 ! INVALID
0180      IF(IANS.GT.0) THEN                                ! POSITIVE RESPONSE
0181          IPRINT = 'Y'                                  ! HARDCOPY
0182      ELSE                                              ! NEGATIVE RESPONSE
0183          IPRINT = 'N'                                  ! NO HARDCOPY
0184          IANS = IANS * -1                               ! MAKE POSITIVE FOR FUTURE US
0185      END IF                                            ! END IF BLOCK
0186
0187      !-----PREDICTION TYPE PERFORMED-----
0188      IF (IANS.EQ.1) THEN                                ! OPERATIONAL SELECTED
0189          400      CALL CLOSE(2)                          ! CLOSE FILE 2
0190          OPEN(UNIT=2,NAME='NEWSVP.DAT;1',TYPE='UNKNOWN', ! OPEN I FILE
0191              1      ACCESS='DIRECT',FORM='UNFORMATTED',RECORDSIZE=340)
0192              IANS=1                                     ! SET IANS
0193              CALL ENVIRN(IPRINT)                       ! UPDATE ENVIRN DATA
0194              CALL CLOSE(2)                              ! CLOSE FILE 2
0195          OPEN(UNIT=2,NAME='NEWSVP.DAT;1',TYPE='UNKNOWN',
0196              1      ACCESS='DIRECT',FORM='UNFORMATTED',RECORDSIZE=340)
0197      ELSE                                              ! FORECASTING SELECTED
0198          450      CALL CLOSE(2)                          ! CLOSE FILE 2
0199          OPEN(UNIT=2,NAME='NWHIST.DAT;1',TYPE='UNKNOWN', ! OPEN FILE
0200              1      ACCESS='DIRECT',FORM='UNFORMATTED',RECORDSIZE=340)
0201              IANS=2                                     ! SET IANS
0202              CALL FORCST(IPRINT)                       ! UPDATE FORCST DATA
0203              CALL CLOSE(2)                              ! CLOSE FILE 2
0204          OPEN(UNIT=2,NAME='NWHIST.DAT;1',TYPE='UNKNOWN', ! OPEN FILE
0205              1      ACCESS='DIRECT',FORM='UNFORMATTED',RECORDSIZE=340)
0206      END IF                                            ! END IF BLOCK
0207
0208      !-----MENU SELECTION-----
0209      50      CALL ICLR                                    ! CLEAR SCREEN
0210      WRITE(5,1100)                                       ! PROMPT FOR MENU CHOICE
0211      WRITE(5,1150)                                       ! PROMPT FOR MENU CHOICE
0212      READ(5,1200) ICHC                                  ! READ OPERATOR CHOICE
0213      IF(ICHC.LT.-11.OR.ICHC.GT.11.OR.ICHC.EQ.0) GO TO 50 ! INVALID
0214      IF(ICHC.GT.0) THEN                                  ! POSITIVE RESPONSE
0215          IPRINT = 'Y'                                    ! HARDCOPY
0216      ELSE                                              ! NEGATIVE RESPONSE
0217          IPRINT = 'N'                                    ! NO HARDCOPY
0218          ICHC = ICHC * -1                                ! MAKE POSITIVE FOR FUTURE US
0219      END IF                                            ! END IF BLOCK
0220
0221      !-----PERFORM MENU SELECTION-----
0222      IF (ICHC.EQ.1) CALL ACTV26(IPRINT) ! SQS-26 PRED/EQPT SETTINGS
0223      IF (ICHC.EQ.2) CALL ACTVLP(IPRINT) ! LAMPS PERF PRED (ACTIVE)
0224      IF (ICHC.EQ.3) CALL VDS(IPRINT)   ! AN/SQS-35 VDS (ACTIVE)
0225      IF (ICHC.EQ.4) CALL OTHERS(IPRINT) ! OTHER UNITS
0226      IF (ICHC.EQ.5) CALL PBB(IPRINT)   ! PASSIVE BROADBAND (ALL)
0227      IF (ICHC.EQ.6) CALL PSSV(IPRINT)  ! PASSIVE NARROWBAND (ALL)
0228      IF (ICHC.EQ.7) GO TO 400           ! UPDATE & DO OPERATIONAL
0229      IF (ICHC.EQ.8) GO TO 450           ! UPDATE & DO FORECASTING
0230      IF (ICHC.EQ.9) CALL SVPTRC         ! RAYTRACE ROUTINE
0231      IF (ICHC.EQ.10) CALL RAXIN(IPRINT) ! RAYMODE
0232      IF (ICHC.NE.11) THEN               ! IF NOT EXIT THEN
0233          GO TO 50                       ! RETURN TO MENU CHOICES
0234      ELSE                               ! IF CHOICE WAS 11
0235          CALL CLOSE(2)                  ! CLOSE FILE 2
0236          STOP 'EXIT SIMAS'             ! EXIT SIMAS

```

```

0237          END IF                                ! END IF BLOCK
0238
0239          !-----FORMAT STATEMENTS-----
0240          900      FORMAT(' SIMAS (SONAR IN-SITU MODE ASSESSMENT SYSTEM)'
0241                    1      ////,4X,'SELECT PREDICTION TYPE DESIRED:'
0242                    2      //,4X,'1 = OPERATIONAL'
0243                    3      //,4X,'2 = FORECASTING'
0244                    4      ///5X,'**** ENTER YOUR CHOICE ****',T60,' ',)
0245          1100     FORMAT(' SIMAS (SONAR IN-SITU MODE ASSESSMENT SYSTEM)'/
0246                    1 ///,4X,'1 = AN/SQS-26 PREDICTIONS/EQUIPMENT SETTINGS (ACTIVE)'
0247                    2 //,4X,'2 = LAMPS PERFORMANCE PREDICTIONS (ACTIVE)'
0248                    3 //,4X,'3 = AN/SQS-35 VDS (ACTIVE)'
0249                    4 //,4X,'4 = OTHER UNITS'
0250                    5 //,4X,'5 = PASSIVE BROADBAND (ALL SYSTEMS)'
0251                    6 //,4X,'6 = PASSIVE NARROWBAND (ALL SYSTEMS)')
0252          1150     FORMAT(
0253                    1 /,4X,'7 = UPDATE OPERATIONAL DATA & DO OPERATIONAL PREDICTIONS'
0254                    2 //,4X,'8 = UPDATE FORECASTING DATA & DO FORECASTING PREDICTIONS'
0255                    3 //,4X,'9 = RAYTRACE ROUTINE'
0256                    4 //,3X,'10 = RAYMODE'
0257                    5 //,3X,'11 = EXIT SIMAS'///
0258                    6 5X,'**** ENTER YOUR CHOICE ****',T60,' ',)
0259          1200     FORMAT(I3)
0260
0261          END

```

```

0001      SUBROUTINE SMOOTH(NBT,NDLYR)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: SMOOTH
0005      ! AUTHOR: STEPHEN LAFLEUR, CODE 3333, NUSC/NLL
0006      ! DATE: 18 SEP 84
0007      ! FUNCTION: SUBROUTINE SMOOTH WILL SMOOTH THE XBT DATA
0008      !              WITHOUT MODIFYING THE LAYER DEPTH POINT.
0009      ! INPUTS: PARAMETERS PASSED IN AND VARIABLES IN COMMONS.
0010      ! OUTPUTS: SMOOTHED XBT DATA
0011      ! MODULES CALLED: NONE
0012      ! CALLED BY: XBT
0013      !
0014      INCLUDE 'DTV.INC'
0015 1 !-----DTV-----
0016 1 ! VARBL   SIZE   PURPOSE                                TYPE   RANGE
0017 1 ! -----
0018 1 ! D       (25)   DEPTH                                    REAL*4
0019 1 ! DD      (25)   DEPTH                                    REAL*4
0020 1 ! NNBT                                NUMBER OF BATHETHERMAL  INTEGER*2
0021 1 ! T       (25)   TEMPERATURE                             REAL*4
0022 1 ! TT      (25)   TEMPERATURE                             REAL*4
0023 1 ! VEL     (25)   VELOCITY                                REAL*4
0024 1 !
0025 1      INTEGER*2 NNBT
0026 1      REAL*4 D,DD,T,TT,VEL
0027 1
0028 1      COMMON /DTV/ D(25),T(25),VEL(25),DD(25),TT(25),NNBT
0029 1 !-----END DTV-----
0030 !
0031 ! VARBL   SIZE   PURPOSE                                TYPE   RANGE
0032 ! -----
0033 ! D1                                DIFFERENCE IN DEPTHS  REAL*4
0034 ! D2                                DIFFERENCE IN DEPTHS  REAL*4
0035 ! V0                                INTERPOLATED SOUND SPEED REAL*4
0036 ! V1                                DELTA SOUND SPEED      REAL*4
0037 ! V2                                DELTA SOUND SPEED      REAL*4
0038 ! VEL1    (25)  VELOCITY                                REAL*4
0039 ! I                                COUNTER                  INTEGER*2
0040 ! J                                COUNTER                  INTEGER*2
0041 ! NBT                                NUMBER OF BT POINTS  INTEGER*2
0042 ! NDLYR                                BT LAYER'S POSITION IN ARRAY  INTEGER*2
0043 !
0044 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMON ***
0045
0046      INTEGER*2 I,J,NBT,NDLYR
0047      REAL*4 D1,D2,VEL1,V0,V1,V2
0048      DIMENSION VEL1(25)
0049
0050 1-----SMOOTH XBT DATA-----
0051      DO 320 I=2,NBT-1      ! SMOOTH MODIFIED DATA
0052      D1=D(I)-D(I-1)        ! DEPTH DIFF THIS DEPTH & NEXT
0053      D2=D(I+1)-D(I)        ! DEPTH DIFF THIS & PREVIOUS
0054      V1=VEL(I-1)-VEL(I)    ! SOUND SPEED DIFF THIS & PREVIOUS
0055      V2=VEL(I+1)-VEL(I)    ! SOUND SPEED DIFF THIS & NEXT
0056      IF(D1.LE.D2) THEN     ! PREVIOUS < NEXT DEPTH
0057      V0=VEL(I)+V2*D1/D2    ! INTERPOLATE SOUND SPEED
0058      VEL1(I)=(VEL(I-1)+VEL(I)+V0)/3 ! SMOOTHED SOUND SPEED
0059      ELSE                  ! PREVIOUS > NEXT DEPTH

```



```

0060          V0=VEL(I)-V1*D2/D1          ! INTERPOLATE SOUND SPEED
0061          VEL1(I)=(V0+VEL(I)+VEL(I+1))/3 ! SMOOTHED SOUND SPEED
0062          END IF                      ! END IF BLOCK
0063          IF(I.EQ.NDLYR) VEL1(I)=VEL(I) ! LEAVE LAYER POINT
0064          320      CONTINUE          ! END DO LOOP
0065          DO 350 J=2,NBT-1          ! BEGIN DO LOOP
0066          VEL(J)=VEL1(J)            ! INSERT SMOOTHED DATA
0067          350      CONTINUE          ! END DO LOOP
0068          RETURN                    ! BACK TO CALLING ROUTINE
0069          END                      ! END SUBROUTINE

```

#### COMMAND QUALIFIERS

```

FORTRAN /CHECK=ALL/LIST/SHOW=(INCLUDE,NOMAP) [LAFLEUR]SMOOTH.F77

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE_FORM)
/SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)
/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD_LINES /NOCROSS_REFERENCE

```

#### COMPILATION STATISTICS

```

Run Time:          1.22 seconds
Elapsed Time:      1.91 seconds
Page Faults:       333
Dynamic Memory:    126 pages

```

```

0001          SUBROUTINE SSP(INSSP)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: SSP
0005      ! AUTHOR: S. LAFLEUR & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 5/84 & 5/84 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE SSP IS USED FOR MANUAL ENTRY OF SSP DATA
0008      !              (DEPTH AND VELOCITY VALUES).
0009      ! INPUTS:  OPERATOR INPUT OF DATA.  VARIABLES IN COMMONS.
0010      ! OUTPUTS: CRT PROMPTING MESSAGES TO OPERATOR.
0011      ! MODULES CALLED: EDITBT,ICLR,METRIC
0012      ! CALLED BY: KEYPCH
0013      !
0014      INCLUDE 'DTV.INC'
0015  1 ! -----DTV-----
0016  1 !  VARBL   SIZE   PURPOSE                                TYPE      RANGE
0017  1 !  -----
0018  1 !  D       (25)   DEPTH                                REAL*4
0019  1 !  DD      (25)   DEPTH                                REAL*4
0020  1 !  NNBT                                NUMBER OF BATHETHERMAL  INTEGER*2
0021  1 !  T       (25)   TEMPERATURE                          REAL*4
0022  1 !  TT      (25)   TEMPERATURE                          REAL*4
0023  1 !  VEL     (25)   VELOCITY                              REAL*4
0024  1 !
0025  1      INTEGER*2 NNBT
0026  1      REAL*4 D,DD,T,TT,VEL
0027  1
0028  1      COMMON /DTV/ D(25),T(25),VEL(25),DD(25),TT(25),NNBT
0029  1 ! -----END DTV-----
0030      INCLUDE 'SVP.INC'
0031  1 ! -----SVP-----
0032  1 !  VARBL   SIZE   PURPOSE                                TYPE      RANGE
0033  1 !  -----
0034  1 !  BDF                                BOTTOM DEPTH IN FATHOMS  REAL*4
0035  1 !  BIOP                                BIOLOGICAL BACK SCATTERING COEF  REAL*4
0036  1 !  BTDATE (9)   DATE OF LAST BT INPUT                    BYTE
0037  1 !  BTTIME (8)   TIME OF LAST BT INPUT                     BYTE
0038  1 !  C       (50)   VELOCITY (PAIRED WITH Z FOR SVP)        REAL*4
0039  1 !  CC      (50)   VELOCITY (PAIRED WITH ZZ FOR SVP)      REAL*4
0040  1 !  CS                                SOUND VELOCITY AT SURFACE  REAL*4
0041  1 !  DEG                                TEMPERATURE (DEG)        REAL*4      57.2957795
0042  1 !  EL                                LAYER DEPTH            DATA
0043  1 !  F                                FREQUENCY                REAL*4
0044  1 !  GRDS                                GRIDS                  REAL*4      0.0164
0045  1 !  ITO                                MINIMAL 2-WAY TRAVEL TIME  INTEGER*2
0046  1 !  MGSOP                                MGS PROVINCE NUMBER      INTEGER*2
0047  1 !  N                                # OF DEPTH/VELOCITY PAIRS  INTEGER*2
0048  1 !  NN                                # OF DEPTH/VELOCITY PAIRS  INTEGER*2
0049  1 !  PI                                MATHEMATICAL CONSTANT PI  REAL*4      3.1415927
0050  1 !  SNDATE (9)   DATE SYS PARMS LAST UPDATED              BYTE
0051  1 !  SNTIME (8)   TIME SYS PARMS LAST UPDTAED               BYTE
0052  1 !  SYDATE (9)   CURRENT DATE READ FROM SYSTEM             BYTE
0053  1 !  SYTIME (8)   CURRENT TIME READ FROM SYSTEM             BYTE
0054  1 !  TMP                                TEMPERATURE              REAL*4
0055  1 !  UMKZ                                BOTTOM BACK SCATTERING COEF.  REAL*4      -28.0
0056  1 !  WS                                WIND SPEED              REAL*4
0057  1 !  Z       (50)   DEPTH OF POINT OF SOUND SPEED           REAL*4
0058  1 !  ZZ      (50)   DEPTH OF POINT OF SOUND SPEED           REAL*4
0059  1

```

```

0060 1      INTEGER*2 ITO,MGSOP,N,NN
0061 1      REAL*4    BDF,BIOP,C(50),CC(50),CS,DEG,EL,F,GRDS
0062 1      REAL*4    PI,TMP,UMKZ,WS,Z(50),ZZ(50)
0063 1      BYTE      SYDATE(9),SYTIME(8),BTDATE(9),BTTIME(8)
0064 1      BYTE      SNDATE(9),SNTIME(8)
0065 1      DATA     PI,DEG,GRDS/3.1415927,57.2957795,0.0164/
0066 1      DATA     UMKZ/-28./
0067 1
0068 1      COMMON /SVP/ F,N,Z,C,EL,MGSOP,BDF,WS,CS,TMP,BIOP,
0069 1          1      UMKZ,PI,DEG,GRDS,ITO,ZZ,CC,NN,
0070 1          2      SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME
0071 1 !-----SVP-END-----
0072      INCLUDE 'SVP1.INC'
0073 1 !-----SVP1-----
0074 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0075 1 ! -----
0076 1 ! BUFFER (224)  HISTORICAL DATA FILE BUFFER          REAL*4
0077 1 ! DS      (30)  HISTORICAL DEPTH                      REAL*4
0078 1 ! J20        # OF DEEP OCEAN DEPTH/VEL PAIRS          INTEGER*2
0079 1 ! NS        TOTAL # OF PAIRS IN HISTORICAL            INTEGER*2
0080 1 ! NSN       MONTH NUMBER (1=JAN.,ETC)                 INTEGER*2  1 TO 12
0081 1 ! SLNTY     SALINITY                                    REAL*4
0082 1 ! VS      (30)  HISTORICAL VELOCITY                  REAL*4
0083 1
0084 1      REAL*4    BUFFER,DS,SLNTY,VS
0085 1      INTEGER*2 J20,NSN,NS
0086 1
0087 1      COMMON /SVP1/ J20,BUFFER(224),NSN,SLNTY,DS(30),VS(30),NS
0088 1 !-----END SVP1-----
0089 1
0090 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0091 1 ! -----
0092 1 ! I      COUNTER                                         INTEGER*2
0093 1 ! IERROR ERROR FLAG FORM METRIC                        INTEGER*2
0094 1 ! INSSP  SSP TYPE SELECTED BY OPERATOR                 INTEGER*2
0095 1 ! J      COUNTER                                         INTEGER*2
0096 1 ! JANS   OPERATOR RESPONSE FOR LAST SSP               INTEGER*2
0097 1 ! L      OPERATOR RESPONSE FOR EDIT SSP               INTEGER*2  Y OR N
0098 1
0099 1 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0100
0101      INTEGER*2 I,IERROR,INSSP,J,JANS,L
0102
0103 10      IERROR = 0 ! SET METRIC ERROR FLAG
0104      CALL ICLR ! CLEAR SCREEN
0105      WRITE(5,2050) ! USE LAST SSP PROMPT
0106      READ(5,1050) JANS ! OPERATOR RESPONSE
0107      IF(JANS.EQ.'Y') THEN ! ****USE PROFILE SAVED
0108          DO 250 I=1,NN ! FROM LAST RUN ****
0109              Z(I)=ZZ(I) ! STORE DEPTH
0110              C(I)=CC(I) ! STORE VELOCITY
0111 250      CONTINUE ! END DO LOOP
0112          N=NN ! NUMBER OF SSP POINTS
0113          GOTO 75 ! GO TO OUTPUT
0114      END IF ! END IF BLOCK
0115 40      CALL ICLR ! CLEAR SCREEN
0116      WRITE(5,2250) ! INPUT SSP
0117      DO 50 J=1,50 ! DO 50 TIMES
0118          WRITE(5,1310) J ! WRITE LOOP COUNTER

```

```

0119      READ(5,1300) Z(J),C(J)          ! READ DEPTH & TEMP
0120      IF(J.GT.1.AND.Z(J).LE.1.)GO TO 60 ! CHECK FOR LAST ENTRY
0121      IF(C(J).LE.100.) GOTO 40         ! INVALID LOOP BACK
0122      50      CONTINUE                ! END DO LOOP
0123      J=51                            ! SET COUNTER TO 51
0124      60      N=J-1                   ! # OF SSP = COUNTER - 1
0125      75      WRITE(5,1380)            ! INPUT DATA TITLE PROMPT
0126      WRITE(5,1400)                   ! PARAMETER TITLES
0127      WRITE(5,1420) (I,Z(I),C(I),I=1,N) ! DEPTH AND TEMP OR SS
0128      WRITE(5,1500)                   ! CHECK ENTRIES FOR ERRORS
0129      READ(5,1050) L                   ! EDIT DATA RESPONSE
0130      IF(L.EQ.'Y') CALL EDITBT(INSSP,N,Z,C) ! CORRECT SSP DATA
0131      DO 100 I=1,N                    ! **** SAVE PROFILE ****
0132      ZZ(I)=Z(I)                      ! STORE DEPTH
0133      CC(I)=C(I)                      ! STORE VELOCITY
0134      100     CONTINUE                 ! END DO LOOP
0135      NN=N                             ! NUMBER OF SSP POINTS
0136      CALL METRIC(INSSP,ZZ,CC,N,Z,C,SLNTY,VS(1),IERROR) ! METRIC CALC
0137      IF(IERROR.EQ.1) GO TO 10         ! ERROR IN DATA INPUT
0138      BDF=Z(N)/6.                      ! BOTTOM DEPTH IN FATHOMS
0139      RETURN                           ! RETURN TO CALLING ROUTINE
0140
0141      !-----FORMAT STATEMENTS-----
0142      1050     FORMAT(A1)
0143      1300     FORMAT(2F10.2)
0144      1310     FORMAT(T5,I5,T22,'****',T32,$)
0145      1380     FORMAT(1H /T26,'OPERATOR INPUT DATA')
0146      1400     FORMAT(//T22,'NO.',T32,'DEPTH',T42,
0147      1         'SOUND'/T42,'SPEED'/)
0148      1420     FORMAT(T23,I2,T32,F7.1,T42,F6.1)
0149      1500     FORMAT(1H0/1H$,4X,'DO YOU WISH TO EDIT THE DATA? YES OR NO',T60)
0150      2050     FORMAT(' DO YOU WANT TO USE THE LAST PROFILE? ',,$)
0151      2250     FORMAT(T20,'ENTER SOUND SPEED PROFILE (50 POINTS MAX)'
0152      1         /T20,'      IN METRIC AND/OR ENGLISH UNITS'
0153      2         /T20,'      (AN EXTRA <CR> TERMINATES ENTRIES)'
0154      3         //T32,'DEPTH',T42,'SOUND'/T42,'SPEED'/)
0155      END

```

17-Dec-1984 15:05:59  
17-Dec-1984 15:05:59

# SUBROUTINE SVPGRF(INPBDP)

```

0001  D1
0002
0003  ! PROLOGUE:
0004  ! MODULE NAME: SVPGRF
0005  ! AUTHOR: JOHN VALLEY, S. LAFLEUR, & W. WACHTER, CODE 3333, NUSC/NLL
0006  ! DATE: 1977, 1982 (REDESIGN), & 11/83 (FORTRAN 77)
0007  ! FUNCTION: SUBROUTINE SVPGRF PRODUCES A HARDCOPY GRAPHIC OUTPUT
0008  !             OF DEEP AND SHALLOW SVPS.
0009  ! INPUTS:   HARD COPY SELECTION.  PARAMETERS PASSED IN.  VARIABLES
0010  !             IN COMMONS.
0011  ! OUTPUTS:  CRT PROMPTING MESSAGES TO OPERATOR
0012  ! MODULES CALLED: ICLR,INSERT,OUTPUT
0013  ! CALLED BY: ENVIRN,FORCST
0014  !
0015  ! =====
0016  ! ALGORITHMS USED:
0017  !
0018  !     MINIMUM SOUND SPEED FOR AXIS LABEL = INTEGER*2 VALUE OF
0019  !     (ACTUAL MINIMUM SPEED - 10) / 50
0020  !
0021  ! =====
0022  !
0023  !     INCLUDE 'DHST.INC'
0024  1 ! -----DHST-----
0025  1 !  VARBL  SIZE  PURPOSE                                TYPE      RANGE
0026  1 !  -----
0027  1 !  SCHNLD          SOUND CHANNEL LAYER DEPTH  REAL*4
0028  1 !
0029  1 !          REAL*4  SCHNLD
0030  1 !
0031  1 !          COMMON /DHST/ SCHNLD
0032  1 ! -----DHST END-----
0033  !     INCLUDE 'ENVN.INC'
0034  1 ! -----ENVN-----
0035  1 !  VARBL  SIZE  PURPOSE                                TYPE      RANGE
0036  1 !  -----
0037  1 !  BIO      (2)    BIOLOGICAL BACK SCATTERING  REAL*4      -57. & -47.
0038  1 !  DLYR          LAYER DEPTH                    REAL*4
0039  1 !  MGS          MGS PROVINCE                    INTEGER*2
0040  1 !
0041  1 !          REAL*4  BIO,DLYR
0042  1 !          INTEGER*2 MGS
0043  1 !          DATA BIO/-57.,-47./
0044  1 !
0045  1 !          COMMON /ENVN/ BIO(2),DLYR,MGS
0046  1 !
0047  1 ! -----END ENVN-----
0048  !     INCLUDE 'GRF.INC'
0049  1 ! -----GRF-----
0050  1 !  VARBL  SIZE  PURPOSE                                TYPE      RANGE
0051  1 !  -----
0052  1 !  DBT      (25)   DEPTH OF DEPTH/VEL PAIR        REAL*4
0053  1 !  IANS          PREDICTION TYPE                  INTEGER*2  -2 TO +2
0054  1 !  ILYR          INDEX FOR LAYER DEPTH            INTEGER*2
0055  1 !  INBT          OPERATOR ENTERED # OF BT POINTS  INTEGER*2
0056  1 !  ISVP          LATEST OR HISTORICAL BT FLAG      INTEGER*2  1 OR 2
0057  1 !  I2000        SVP INDEX FOR 2000 FT DEPTH        INTEGER*2

```

C-48.1

```

0058 1 ! VBT      (25)  VELOCITY FOR DEPTH PAIR REAL*4      REAL*4
0059 1
0060 1      REAL*4      DBT,VBT
0061 1      INTEGER*2  IANS,ILYR,INBT,ISVP,I2000
0062 1
0063 1      COMMON /GRF/ IANS,ISVP,ILYR,I2000,INBT,DBT(25),VBT(25)
0064 1
0065 1 ! -----END GRF-----
0066      INCLUDE 'LOC.INC'
0067 1 ! -----LOC-----
0068 1 !  VARBL  SIZE      PURPOSE                      TYPE      RANGE
0069 1 !  -----
0070 1 !  INDX      SSP INDEX                      INTEGER*2
0071 1 !  LAT      (4)    LATITUDE                      INTEGER*2
0072 1 !  LONG      (4)    LONGITUDE                      INTEGER*2
0073 1 !  NMAREA   (20)    AREA OCEAN NAME                      BYTE
0074 1 !  NOC      NUMBER OF OCEAN                      INTEGER*2
0075 1 !  RCZ      RANGE TO CONVERG. ZONE REAL*4
0076 1
0077 1      REAL*4      RCZ
0078 1      INTEGER*2  INDX,LAT,LONG,NOC
0079 1      BYTE      NMAREA(20)
0080 1
0081 1      COMMON /LOC/ LAT(4),LONG(4),NOC,INDX,RCZ,NMAREA
0082 1
0083 1 ! -----END LOC-----
0084      INCLUDE 'OCEANS.INC'
0085 1
0086 1 ! -----OCEANS-----
0087 1 !  VARBL  SIZE      PURPOSE                      TYPE
0088 1 !  -----
0089 1 !  IOCEAN (50)    ARRAY OF NAMES OF OCEANS                      DATA
0090 1
0091 1      INTEGER*2  IOCEAN
0092 1      DIMENSION IOCEAN(50)
0093 1
0094 1      DATA IOCEAN/'NO','RT','H ','PA','CI','FI','C ','OC','EA','N ',
0095 1      1  'NO','RT','H ','AT','LA','NT','IC','O','CE','AN',
0096 1      2  'ME','DI','TE','RR','AN','EA','N ','SE','A ',
0097 1      3  'IN','DI','AN','O','CE','AN',' ',' ',' ',' ',
0098 1      4  'NO','RW','EG','IA','N ','SE','A ',
0099 1
0100 1      COMMON /OCEANS/ IOCEAN
0101 1
0102 1 ! -----END OCEANS-----
0103      INCLUDE 'SVP.INC'
0104 1 ! -----SVP-----
0105 1 !  VARBL  SIZE      PURPOSE                      TYPE      RANGE
0106 1 !  -----
0107 1 !  BDF      BOTTOM DEPTH IN FATHOMS                      REAL*4
0108 1 !  BIOP     BIOLOGICAL BACK SCATTERING COEF              REAL*4
0109 1 !  BTDATE   (9)    DATE OF LAST BT INPUT                  BYTE
0110 1 !  BTTIME   (8)    TIME OF LAST BT INPUT                   BYTE
0111 1 !  C        (50)    VELOCITY (PAIRED WITH Z FOR SVP)        REAL*4
0112 1 !  CC        (50)    VELOCITY (PAIRED WITH ZZ FOR SVP)      REAL*4
0113 1 !  CS      SOUND VELOCITY AT SURFACE                      REAL*4
0114 1 !  DEG      TEMPERATURE (DEG)                              REAL*4      57.2957795

```

```

15 1 ! EL          LAYER DEPTH          DATA
0116 1 ! F          FREQUENCY          REAL*4
0117 1 ! GRDS       GRIDS              REAL*4      0.0164
0118 1 ! ITO        MINIMAL 2-WAY TRAVEL TIME  INTEGER*2
0119 1 ! MGSOP      MGS PROVINCE NUMBER  INTEGER*2
0120 1 ! N          # OF DEPTH/VELOCITY PAIRS  INTEGER*2
0121 1 ! NN         # OF DEPTH/VELOCITY PAIRS  INTEGER*2
0122 1 ! PI         MATHEMATICAL CONSTANT PI  REAL*4      3.1415927
0123 1 ! SNDATE (9)  DATE SYS PARMS LAST UPDATED  BYTE
0124 1 ! SNTIME (8)  TIME SYS PARMS LAST UPDTAED  BYTE
0125 1 ! SYDATE (9)  CURRENT DATE READ FROM SYSTEM  BYTE
0126 1 ! SYTIME (8)  CURRENT TIME READ FROM SYSTEM  BYTE
0127 1 ! TMP        TEMPERATURE          REAL*4
0128 1 ! UMKZ       BOTTOM BACK SCATTERING COEF.  REAL*4      -28.0
0129 1 ! WS         WIND SPEED            REAL*4
0130 1 ! Z          (50)  DEPTH OF POINT OF SOUND SPEED  REAL*4
0131 1 ! ZZ         (50)  DEPTH OF POINT OF SOUND SPEED  REAL*4
0132 1
0133 1          INTEGER*2 ITO,MGSOP,N,NN
0134 1          REAL*4    BDF,BIOP,C(50),CC(50),CS,DEG,EL,F,GRDS
0135 1          REAL*4    PI,TMP,UMKZ,WS,Z(50),ZZ(50)
0136 1          BYTE      SYDATE(9),SYTIME(8),BTDATE(9),BTTIME(8)
0137 1          BYTE      SNDATE(9),SNTIME(8)
0138 1          DATA     PI,DEG,GRDS/3.1415927,57.2957795,0.0164/
0139 1          DATA     UMKZ/-28./
0140 1
0141 1          COMMON /SVP/ F,N,Z,C,EL,MGSOP,BDF,WS,CS,TMP,BIOP,
0142 1          1          UMKZ,PI,DEG,GRDS,ITO,ZZ,CC,NN,
0143 1          2          SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME
0144 1 ! -----SVP-END-----
0145          INCLUDE 'SVP1.INC'
0146 1 ! -----SVP1-----
0147 1 ! VARBL  SIZE  PURPOSE              TYPE  RANGE
0148 1 ! -----
0149 1 ! BUFFER (224)  HISTORICAL DATA FILE BUFFER  REAL*4
0150 1 ! DS        (30)  HISTORICAL DEPTH          REAL*4
0151 1 ! J20        # OF DEEP OCEAN DEPTH/VEL PAIRS  INTEGER*2
0152 1 ! NS        TOTAL # OF PAIRS IN HISTORICAL  INTEGER*2
0153 1 ! NSN       MONTH NUMBER (1=JAN.,ETC)        INTEGER*2  1 TO 12
0154 1 ! SLNTY     SALINITY                        REAL*4
0155 1 ! VS        (30)  HISTORICAL VELOCITY        REAL*4
0156 1
0157 1          REAL*4    BUFFER,DS,SLNTY,VS
0158 1          INTEGER*2 J20,NSN,NS
0159 1
0160 1          COMMON /SVP1/ J20,BUFFER(224),NSN,SLNTY,DS(30),VS(30),NS
0161 1 ! -----END SVP1-----
0162 1
0163 1 ! VARBL  SIZE  PURPOSE              TYPE  RANGE
0164 1 ! -----
0165 1 ! CMIN      MINIMUM SOUND SPEED              REAL*4
0166 1 ! FTER      CONVERT FEET TO RASTER UNITS      REAL*4
0167 1 ! I         LOOP COUNTER                    INTEGER*2
0168 1 ! IBLNK     BLANK SPACES                     DATA
0169 1 ! IC        (50)  VELOCITY IN RASTERS          INTEGER*2
0170 1 ! ICOPY     NUMBER OF HARDCOPIES REQUESTED BY OPERATOR  INTEGER*2
0171 1 ! IDDAT    (14)  'OPERATION AREA FORECAST AREA' LABEL  DATA

```

```

0172 ! IDF                END POINTER FOR JSPDAT ARRAY                INTEGER*2
0173 ! IDEPSC            SOUND CHANNEL DEPTH                        INTEGER*2
0174 ! IGTYP             GRAPH TYPE DEPENDENT ON BOTTOM DEPTH        INTEGER*2    1-3
0175 ! IJECT            OPERATOR RESPONSE FOR GRAPHIC OUTPUT        INTEGER*2
0176 ! INCI             RASTER DIFF BETWEEN DEPTH LABEL HEIGHTS    INTEGER*2
0177 ! INPBDF           INPUTTED BOTTOM DEPTH IN FATHOMS            INTEGER*2
0178 ! IRCZ            RANGE TO CONVERGENCE ZONE                    INTEGER*2
0179 ! ISTRT           BOTTOM LINE RASTER HT (DEEP PROFILE GRAPH)   INTEGER*2
0180 ! ITOT            TOTAL NUMBER OF DEPTH LINES MINUS ONE        INTEGER*2
0181 ! IUD             STARTING POINTER FOR JSPDAT ARRAY            INTEGER*2
0182 ! IXNOW           X COORDINATE IN RASTERS                      INTEGER*2
0183 ! IXSTRT          LOOP COUNTER                                INTEGER*2
0184 ! IY              FACTOR FOR DEEP SVP HORIZONTALS              REAL*4
0185 ! IYPLOT           Y COORDINATE IN RASTERS                      INTEGER*2
0186 ! IYPOS           Y COORDINATE IN RASTERS                      INTEGER*2
0187 ! IYSPOT          Y COORDINATE IN RASTERS                      INTEGER*2
0188 ! IZ              (50) DEPTH IN RASTERS                        INTEGER*2
0189 ! I6              POINTER FOR IDDAT ARRAY                      INTEGER*2
0190 ! J               LOOP COUNTER                                INTEGER*2
0191 ! JBTFMDP         BOTTOM DEPTH IN FATHOMS                      INTEGER*2
0192 ! JCMAX           MAXIMUM SOUND SPEED ON GRAPHS                INTEGER*2
0193 ! JCMIN           MINIMUM SOUND SPEED ON GRAPHS                INTEGER*2
0194 ! JF             MAXIMUM DEPTH OF THE GRAPH                    INTEGER*2
0195 ! JJ             MAX SOUND SPEED FOR AXIS LABEL                INTEGER*2
0196 ! JNBT           NUMBER OF BATHETHERMALS                      INTEGER*2
0197 ! JS            DEPTH BETWEEN LABELED DEPTHS                  INTEGER*2
0198 ! JSPDAT (30)    'HISTORICAL LATEST XBT KEYPUNCHED LABELS    DATA
0199 ! JWIND           WIND SPEED                                    INTEGER*2
0200 ! K              LOOP COUNTER                                INTEGER*2
0201 ! L              LOOP COUNTER                                INTEGER*2
0202 ! LAYER          LAYER DEPTH                                  INTEGER*2
0203 ! M              LOOP COUNTER                                INTEGER*2
0204 ! NL              (8) LABEL 'DEPTH FT'                          DATA
0205 ! NNN            LOOP COUNTER                                INTEGER*2
0206 ! RMONTH (12)    ARRAY OF NAMES OF MONTHS                      DATA
0207 ! RSTEP          IDEAL RASTER DIFFERENCE BETWEEN LINES        REAL*4
0208 ! SCHMXD         SOUND CHANNEL MAXIMUM DEPTH                  REAL*4
0209 ! THEBD         BOTTOM DEPTH IN FEET                          REAL*4
0210 ! XA            FACTOR FOR SHALLOW SVP VERICALS              REAL*4
0211 ! XX              (50) FACTOR FOR VELOCITY IN RASTERS          REAL*4
0212 ! X1            FACTOR FOR DEEP SVP VERICALS                  REAL*4
0213 ! YA            FACTOR FOR SHALLOW SVP HORIZONTALS            REAL*4
0214 ! YY              (50) FACTOR FOR DEPTH IN RASTERS            REAL*4
0215 ! YI            FACTOR FOR DEEP SVP HORIZONTALS                REAL*4
0216 ! Z2000         BOTTOM DEPTH                                    REAL*4
0217 !
0218 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0219
0220 ! INTEGER*2 IDDAT
0221 ! INTEGER*2 I, IC, ICOPY, IDF, IDEPSC, IGTYP, IJECT, INCI, INPBDF
0222 ! INTEGER*2 IRCZ, ISTRT, ITOT, IUD, IXNOW, IXSTRT, IY, IYPLOT
0223 ! INTEGER*2 IYPOS, IYSPOT, IZ, I6, J, JBTFMDP, JCMAX, JCMIN, JF, JJ
0224 ! INTEGER*2 JNBT, JS, JWIND, K, L, LAYER, M, NNN
0225 ! INTEGER*2 JSPDAT, NL, IBLNK
0226 ! REAL*4 CMIN, FTER, RSTEP, SCHMXD, THEBD, XA, XX, X1, YA, YY, YI, Z2000
0227
0228 ! DIMENSION IC(50), IZ(50), XX(50), YY(50), JSPDAT(30), IDDAT(14)

```



```

0230 DIMENSION NL(8)
0231 DIMENSION RMONTH(12)
0232 DATA RMONTH/'JAN ','FEB ','MAR ','APR ','
0233           1 'MAY ','JUN ','JUL ','AUG ','
0234           2 'SEP ','OCT ','NOV ','DEC '/
0235 DATA NL/'D','E','F','T','H',' ','F','T'/
0236 DATA IDDAT/'OP','ER','AT','IO','N ','AR','EA','FO','RE','CA',
0237 1'ST','A','RE','A '/
0238 DATA JSPDAT/'H','I','S','T','O','R','I','C','A','L',
0239 1'L','A','T','E','S','T',' ','X','B','T',
0240 2'K','E','Y','P','U','N','C','H','E','D'/
0241 DATA IBLNK/' '/
0242 !=====PRELIMINARIES=====
0243 SCHMXD = 1000. ! MAX SOUND CHANNEL DEPTH
0244 Z2000=2000. ! BOTTOM DEPTH
0245 IF(6.*BDF.LT.Z2000) Z2000=6.*BDF! SET MINIMUM BOTTOM DEPTH
0246 CALL INSERT(N,Z,C,Z2000,I2000) ! INSERT DEPTH/VELOCITY POINT
0247
0248 !-----DEEP PROFILE GRAPH TYPE FROM TRUE BOTTOM DEPTH
0249 THEBD=6.0*BDF ! BOTTOM DEPTH IN FEET
0250 IF(THEBD.GE.0..AND.THEBD.LE.12000.) IGTYP=1 ! TYPE 1 <12,000
0251 IF(THEBD.GT.12000..AND.THEBD.LE.16000.) IGTYP=2 ! TYPE 2 <16,000
0252 IF(THEBD.GT.16000.) IGTYP=3 ! TYPE 3 >16,000
0253
0254 !-----INITIALIZE CONVERSION FACTORS
0255 SCHNLD=10000. ! SOUND CHANNEL DEPTH
0256 CMIN=C(1) ! MINIMUM SOUND SPEED
0257 IF(ILYR.LT.N-1) THEN ! LESS THAN NEXT TO LAST
0258 DO 53 I=ILYR+1,N-1 ! LAYER DEPTH+1 TO # OF PAIRS
0259 IF(C(I).LE.C(I+1).AND.SCHNLD.EQ.10000.) SCHNLD=Z(I) ! RESET
0260 IF(C(I).LT.CMIN) CMIN=C(I) ! MINIMUM FOR MIN SOUND SPEED
0261 53 CONTINUE ! END DO LOOP
0262 END IF ! END IF BLOCK
0263 CALL OUTPUT(INPBDF) ! PRINT SVP INFO FOR CHECKING
0264 JJ=IIFIX((CMIN-10.)/50.) ! MAX SOUND SD FOR AXIS LABEL
0265 JCMIN=50*JJ ! MIN SOUND SPEED ON GRAPHS
0266 JCMAX=JCMIN+300 ! MAX SOUND SPEED ON GRAPHS
0267
0268 !-----SET PARMS FOR DRAWING GRAPHS
0269 IF (IGTYP.EQ.1) THEN ! BOTTOM DEPTH < 12,000 FEET
0270 FTER=70./3000. ! CONVERT FEET TO RASTER UNITS
0271 ISTRT=63 ! BOTTOM LINE RASTER HT
0272 INCI=70 ! RASTER DIFF: DEPTH LABEL HTS
0273 RSTEP=70./3. ! RASTER DIFF BETWEEN LINES
0274 ITOT=12 ! TOTAL # OF DEPTH LINES-1
0275 JF=12000 ! MAXIMUM DEPTH OF THE GRAPH
0276 JS=3000 ! DEPTH BETWEEN LABELED DEPTHS
0277 END IF ! END IF BLOCK
0278 IF (IGTYP.EQ.2) THEN ! BOTTOM DEPTH < 16,000 FEET
0279 FTER=70./4000. ! CONVERT FEET TO RASTER UNITS
0280 ISTRT=63 ! BOTTOM LINE RASTER HT
0281 INCI=70 ! RASTER DIFF:DEPTH LABEL HTS
0282 RSTEP=70./4. ! RASTER DIFF BETWEEN LINES
0283 ITOT=16 ! TOTAL # OF DEPTH LINES-1
0284 JF=16000 ! MAX DEPTH OF THE GRAPH
0285 JS=4000 ! DEPTH BETWEEN LABELED DEPTHS

```

```

0286      END IF                                ! END IF BLOCK
0287      IF (IGTYP.EQ.3) THEN                  ! BOTTOM DEPTH > 16,000 FEET
0288          FTER=70./5000.                    ! CONVERT FEET TO RASTER UNITS
0289          ISTRT=49                          ! BOTTOM LINE RASTER HT
0290          INCI=42                            ! RASTER DIFF: DEPTH LABEL HTS
0291          RSTEP=42./3.                      ! RASTER DIFF BETWEEN LINES
0292          ITOT=21                            ! TOTAL # OF DEPTH LINES-1
0293          JF=21000                          ! MAXIMUM DEPTH OF THE GRAPH
0294          JS=3000                           ! DEPTH BETWEEN LABELED DEPTHS
0295      END IF                                ! END IF BLOCK
0296      WRITE(5,106)                          ! PROMPT FOR GRAPHIC OUTPUT
0297      READ(5,950) IJECT                     ! RESPONSE FOR GRAPHICS
0298      IF(IJECT.NE.'Y') GOTO 999             ! NO GRAPHICS WANTED, RETURN
0299
0300      !=====DO GRAPHICS=====
0301      CALL INITT(3)                          ! PREPARE TK4025 FOR GRAPHICS
0302                                          ! 0<=X<=639 0<=Y<=419 RASTERS
0303
0304      !-----DRAW THE DEEP PROFILE-----
0305      DO 110 I=0,ITOT                        ! DO HORIZONTALS
0306          VI=RSTEP*FLOATI(I)                ! # OF RASTERS SO FAR
0307          IY=IIFIX(VI)                      ! INTEGER*2 COUNTERPART
0308          IYPLT=IY+ISTRT                    ! Y COORDINATE IN RASTERS
0309          IF(IY/INCI.EQ.IIFIX(VI)/INCI) THEN ! HORIZONTALS AND TICS
0310              CALL LNTYPE(1)                 ! TYPE OF LINE
0311              CALL CONECT(55,IYPLT,255,IYPLT)! DRAW LINE BETWEEN PTS
0312          ELSE                               ! HORIZONTALS ONLY
0313              CALL LNTYPE(2)                 ! TYPE OF LINE .....
0314              CALL CONECT(63,IYPLT,255,IYPLT)! DRAW LINE BETWEEN PTS
0315          END IF                             ! END IF BLOCK
0316      110  CONTINUE                          ! END DO LOOP
0317
0318      DO 120 K=63,255,32                     ! DO VERTICALS
0319          X1=FLOATI(K)                      ! INTEGER*2 COUNTERPART
0320          IF((FLOATI(K)+1.)/64..NE.(X1+1.)/64.) THEN ! VERTICALS AND TICS
0321              CALL LNTYPE(2)                 ! TYPE OF LINE .....
0322              CALL CONECT(K,ISTRT,K,IYPLT)! DRAW LINE BETWEEN 2 PTS
0323          ELSE                               ! VERTICALS ONLY
0324              CALL LNTYPE(1)                 ! TYPE OF LINE
0325              CALL CONECT(K,ISTRT,K,IYPLT+7)! DRAW LINE BETWEEN 2 PTS
0326          END IF                             ! END IF BLOCK
0327      120  CONTINUE                          ! END DO LOOP
0328
0329      DO 130 K=1,N                           ! .64=RASTERS/(FT/SEC)
0330          XX(K)=.64*(C(K)-FLOATI(JCMIN))+63. ! 63=RASTER COORD. OF JCMIN
0331          YY(K)=343.-(FTER*Z(K))            ! 343=RASTER HEIGHT OF 0 DEPTH
0332          IC(K)=IIFIX(XX(K))                ! VELOCITY IN RASTER UNITS
0333          IZ(K)=IIFIX(YY(K))                ! DEPTH IN RASTER UNITS
0334      130  CONTINUE                          ! END DO LOOP
0335      CALL CONECT(IC(1),IZ(1),IC(2),IZ(2)) ! DRAW LINE BETWEEN PTS
0336
0337      DO 140 M=3,N                           ! FOR NUMBER OF PAIRS
0338          CALL DRAW(IC(M),IZ(M))             ! DRAW DEEP SVP
0339      140  CONTINUE                          ! END DO LOOP
0340
0341      !-----DRAW SHALLOW PROFILE-----
0342      DO 150 M=63,343,14                     ! DO HORIZONTALS

```

```

13      YA=FLOATI(M)                ! REAL COUNTERPART
0344    IF((M-63)/28.EQ.(IIFIX(YA)-63)/28) THEN! HORIZONTALS AND TICS
0345      CALL LNTYPE(1)              ! TYPE OF LINE
0346      CALL CONECT(423,M,623,M)    ! DRAW LINE BETWEEN 2 PTS
0347    ELSE                          ! HORIZONTALS ONLY
0348      CALL LNTYPE(2)              ! TYPE OF LINE
0349      CALL CONECT(431,M,623,M)    ! DRAW LINE BETWEEN 2 PTS
0350    END IF                          ! END IF BLOCK
0351 150    CONTINUE                  ! END DO LOOP
0352    M=IIFIX(YA)                    ! INTEGER*2 COUNTERPART
0353
0354    DO 160 NNN=431,643,32           ! DO VERTICALS
0355      XA=FLOATI(NNN)                ! REAL COUNTERPART
0356      IF((NNN-431)/64.NE.(IIFIX(XA)-431)/64) THEN ! VERTICALS ONLY
0357        CALL LNTYPE(2)              ! TYPE OF LINE
0358        CALL CONECT(NNN,63,NNN,M)   ! DRAW LINE BETWEEN PTS
0359      ELSE                          ! VERTICALS AND TICS
0360        CALL LNTYPE(1)              ! TYPE OF LINE
0361        CALL CONECT(NNN,63,NNN,M+7) ! DRAW LINE BETWEEN PTS
0362      END IF                          ! END IF BLOCK
0363 160    CONTINUE                  ! END DO LOOP
0364
0365    DO 170 NNN=1,I2000              ! CONVERT C AND Z TO RASTER
0366      IC(NNN)=IC(NNN)+368           ! 368=RASTER SHIFT:DEEP-SHALLOW
0367      YY(NNN)=343.-.14*Z(NNN)      ! .14=RASTERS/(FOOT OF DEPTH)
0368      IZ(NNN)=IIFIX(YY(NNN))       ! 343=RASTER HEIGHT OF 0 DEPTH
0369 170    CONTINUE                  ! END DO LOOP
0370    CALL CONECT(IC(1),IZ(1),IC(2),IZ(2))! DRAW LINE BETWEEN 2 PTS
0371
0372    DO 180 NNN=3,I2000              ! DO PROFILE
0373      CALL DRAW(IC(NNN),IZ(NNN))    ! DRAW SHALLOW PROFILE
0374 180    CONTINUE                  ! ! END DO LOOP
0375
0376    ! _____IF RESULTANT PROFILE WAS FORMED BY MERGING WITH AN XBT,
0377    ! _____PLOT THE RAW XBT DATA
0378    IF(INBT.NE.0.AND.ISVP.NE.5) THEN ! NEW XBT GIVEN
0379      WRITE(5,191)                  ! WRITE RAW DATA IN SHALLOW SVF
0380      DO 192 M=1,INBT               ! FOR NUMBER OF BTS
0381        IF(DBT(M).GT.2000.)GOTO 193 ! IF DEPTH OF BT > MAX, OUT
0382        XX(M)=.64*(VBT(M)-FLOATI(JCMIN))+431.0! FACTOR
0383        IC(M)=IIFIX(XX(M))           ! VELOCITY IN RASTERS
0384        YY(M)=343.-.14*DBT(M)       ! FACTOR
0385        IZ(M)=IIFIX(YY(M))           ! DEPTH IN RASTERS
0386 192      CONTINUE                  ! END DO LOOP
0387 193      JNBT=M-1                 ! RESET NUMBER OF BTS
0388      DO 194 K=1,2                 ! DO TWICE
0389        CALL SYMBOL((IC(K)-2),(IZ(K)-2),0,1,'X') ! PLOT SYMBOL
0390 194      CONTINUE                  ! END DO LOOP
0391      IF(JNBT.GE.3) THEN           ! IF NUMBER OF BT >= 3
0392        DO 195 M=3,JNBT            ! DO UNTIL NUMBER OF BTS
0393          CALL SYMBOL((IC(M)-2),(IZ(M)-2),0,1,'X') ! PLOT SYMBOL
0394 195      CONTINUE                  ! END DO LOOP
0395      END IF                        ! END IF BLOCK
0396      CALL LNTYPE(2)                ! TYPE OF LINE
0397      CALL CONECT(IC(1),IZ(1),IC(2),IZ(2)) ! DRAW LINE BETWEEN 2 PTS
0398      CALL LNTYPE(2)                ! TYPE OF LINE
0399      DO 198 M=3,JNBT              ! DO UNTIL NUMBER OF BTS

```

```

b0          CALL DRAW(IC(M),IZ(M))          ! DRAW TO THESE COORDINATES
0401      198      CONTINUE                  ! END DO LOOP
0402          CALL LNTYPE(1)                  ! TYPE OF LINE
0403          END IF                          ! END IF BLOCK
0404
0405      !-----THE SPEEDS OF BOTH GRAPHS-----
0406          DO 205 IXSTRT=48,416,368        ! LOOP COUNTER
0407              IXNOW=IXSTRT                  ! X COORDINATE IN RASTERS
0408              DO 200 L=JCMIN,JCMAX,100      ! FOR ALL SOUND SPEEDS
0409                  CALL MOVE(IXNOW,363)      ! MOVE TO THESE COORDS
0410                  CALL INUMBR(L,4)          ! WRITE NUMBER
0411                  IXNOW=IXNOW+64            ! RESET X COORDINATE
0412      200      CONTINUE                    ! END DO LOOP
0413      205      CONTINUE                    ! END DO LOOP
0414
0415      !-----LABEL THE PROFILE DEPTHS
0416          IYSPOT=349                        ! Y COORDINATE IN RASTERS
0417          DO 210 J=0,JF,JS                  ! LABEL DEEP PROFILE DEPTH
0418              CALL MOVE(16,IYSPOT)          ! MOVE TO THESE COORDS
0419              CALL INUMBR(J,5)              ! WRITE NUMBER
0420              IYSPOT=IYSPOT-INCI            ! RESET Y COORDINATE
0421      210      CONTINUE                    ! END DO LOOP
0422          IYPOS=349                        ! Y COORDINATE IN RASTERS
0423          DO 220 J=0,2000,200                ! LABEL SHALLOW PROFILE DEPTH
0424              CALL MOVE(392,IYPOS)          ! MOVE TO THESE COORDS
0425              CALL INUMBR(J,4)              ! WRITE NUMBER
0426              IYPOS=IYPOS-28                ! RESET Y COORDINATE
0427      220      CONTINUE                    ! END DO LOOP
0428
0429      !-----PUT THE ACTUAL SVP DATA BETWEEN THE GRAPHS
0430          CALL MOVE(264,343)                ! MOVE TO THESE COORDS
0431          CALL TEXT(12,'DEPTH  SPEED')      ! WRITE TEXT STRING
0432          IYPOS=329                          ! Y COORDINATE IN RASTER
0433          M=I2000                            ! SET M TO I2000
0434          IF(N.LE.21) M=N                    ! RESET M IF # OF PAIRS<=21
0435          DO 230 K=1,M                      ! DO FOR MAX OF 21 ENTRIES
0436              IZ(K)=IIFIX(Z(K))              ! DEPTH IN RASTERS
0437              IC(K)=IIFIX(C(K))              ! VELOCITY IN RASTERS
0438              CALL MOVE(264,IYPOS)          ! MOVE TO THESE COORDS
0439              CALL INUMBR(IZ(K),5)          ! WRITE NUMBER
0440              CALL MOVE(320,IYPOS)          ! MOVE TO THESE COORDS
0441              CALL INUMBR(IC(K),4)          ! WRITE NUMBER
0442              IYPOS=IYPOS-14                ! RESET Y COORDINATE
0443      230      CONTINUE                    ! END DO LOOP
0444          IF(M.EQ.N) GOTO 255                ! NUMBER OF PAIRS <= 21
0445          DO 240 M=1,3                      ! DO THREE TIMES
0446              CALL CONECT(287,IYPOS+2,287,IYPOS)! DRAW LINE BETWEEN PTS
0447              CALL CONECT(336,IYPOS+2,336,IYPOS)! DRAW LINE BETWEEN PTS
0448              IYPOS=IYPOS-7                ! RESET Y COORDINATE
0449      240      CONTINUE                    ! END DO LOOP
0450          IYPOS=IYPOS-7                      ! RESET Y COORDINATE
0451          NNN=N-(18-I2000)                  ! RESET NNN
0452          IF(NNN.GT.N) GOTO 255              ! IF NNN > NUMBER OF PAIRS
0453          DO 250 L=NNN,N                    ! DO UNTIL NUMBER OF PAIRS
0454              IZ(L)=IIFIX(Z(L))              ! DEPTH IN RASTERS
0455              IC(L)=IIFIX(C(L))              ! VELOCITY IN RASTERS
0456              CALL MOVE(264,IYPOS)          ! MOVE TO THESE COORDS

```

```

    57          CALL INUMBR(IZ(L),5)          ! WRITE NUMBER
0458          CALL MOVE(320,IYPOS)          ! MOVE TO THESE COORDS
0459          CALL INUMBR(IC(L),4)          ! WRITE NUMBER
0460          IYPOS=IYPOS-14                ! RESET Y COORDINATE
0461          250      CONTINUE              ! END DO LOOP
0462          255      CONTINUE              ! END DO LOOP
0463
0464      !-----PUT ON THE TOP LINE OF TEXT-----
0465          CALL MOVE(0,413)                ! MOVE TO THESE COORDS
0466          CALL TEXT(35,'DATE AND TIME OF IN-SITU UPDATE IN ')!WRITE TEXT
0467          I6=(IANS-1)*7+1                ! POINTER FOR IDDAT ARRAY
0468          CALL MOVE(280,413)              ! MOVE TO THESE COORDS
0469          CALL TEXT(14,IDDAT(I6))         ! WRITE TEXT STRING
0470          CALL MOVE(424,413)              ! MOVE TO THESE COORDS
0471          IF(IANS.NE.2) THEN              ! OPERATION PREDICTION TYPE
0472              CALL TEXT(1,BTDATE(1))      ! WRITE TEXT STRING
0473              CALL MOVE(432,413)          ! MOVE TO THESE COORDS
0474              CALL TEXT(1,BTDATE(2))      ! WRITE TEXT STRING
0475              CALL MOVE(440,413)          ! MOVE TO THESE COORDS
0476              CALL TEXT(1,BTTIME(1))      ! WRITE TEXT STRING
0477              CALL MOVE(448,413)          ! MOVE TO THESE COORDS
0478              CALL TEXT(1,BTTIME(2))      ! WRITE TEXT STRING
0479              CALL MOVE(456,413)          ! MOVE TO THESE COORDS
0480              CALL TEXT(1,BTTIME(4))      ! WRITE TEXT STRING
0481              CALL MOVE(464,413)          ! MOVE TO THESE COORDS
0482              CALL TEXT(1,BTTIME(5))      ! WRITE TEXT STRING
0483              CALL MOVE(472,413)          ! MOVE TO THESE COORDS
0484              CALL TEXT(2,'Z ')           ! WRITE TEXT STRING
0485              CALL MOVE(488,413)          ! MOVE TO THESE COORDS
0486              CALL TEXT(1,BTDATE(4))      ! WRITE TEXT STRING
0487              CALL MOVE(496,413)          ! MOVE TO THESE COORDS
0488              CALL TEXT(1,BTDATE(5))      ! WRITE TEXT STRING
0489              CALL MOVE(504,413)          ! MOVE TO THESE COORDS
0490              CALL TEXT(1,BTDATE(6))      ! WRITE TEXT STRING
0491              CALL MOVE(512,413)          ! MOVE TO THESE COORDS
0492              CALL TEXT(1,' ')            ! WRITE TEXT STRING
0493              CALL MOVE(520,413)          ! MOVE TO THESE COORDS
0494              CALL TEXT(1,BTDATE(8))      ! WRITE TEXT STRING
0495              CALL MOVE(528,413)          ! MOVE TO THESE COORDS
0496              CALL TEXT(1,BTDATE(9))      ! WRITE TEXT STRING
0497          ELSE                            ! FORECAST PREDICTION TYPE
0498              CALL TEXT(4,RMONTH(NSN))    ! WRITE TEXT STRING
0499          END IF                          ! END IF BLOCK
0500
0501      !-----PUT ON THE SECOND LINE OF TEXT-----
0502          IF(ISVP.EQ.5) THEN              ! NUMBER OF OCEAN = -1
0503              CALL MOVE(24,399)            ! MOVE TO THESE COORDS
0504              CALL TEXT(20,NMAREA)        ! WRITE TEXT STRING
0505          END IF                          ! END IF BLOCK
0506          IF(NOC.EQ.1) THEN               ! NUMBER OF OCEAN = 1
0507              CALL MOVE(32,399)            ! MOVE TO THESE COORDS
0508              CALL TEXT(20,IOCEAN)        ! WRITE TEXT STRING
0509          END IF                          ! END IF BLOCK
0510          IF(NOC.EQ.2) THEN               ! NUMBER OF OCEAN = 2
0511              CALL MOVE(24,399)            ! MOVE TO THESE COORDS
0512              CALL TEXT(20,IOCEAN(11))    ! WRITE TEXT STRING
0513          END IF                          ! END IF BLOCK

```

```

)14      IF(NOC.EQ.3) THEN      ! NUMBER OF OCEAN = 3
0515      CALL MOVE(48,399)      ! MOVE TO THESE COORDS
0516      CALL TEXT(20,IOCEAN(21)) ! WRITE TEXT STRING
0517      END IF                ! END IF BLOCK
0518      IF(NOC.EQ.4) THEN      ! NUMBER OF OCEAN = 4
0519      CALL MOVE(88,399)      ! MOVE TO THESE COORDS
0520      CALL TEXT(20,IOCEAN(31)) ! WRITE TEXT STRING
0521      END IF                ! END IF BLOCK
0522      IF(NOC.EQ.5) THEN      ! NUMBER OF OCEAN = 5
0523      CALL MOVE(80,399)      ! MOVE TO THESE COORDS
0524      CALL TEXT(20,IOCEAN(41)) ! WRITE TEXT STRING
0525      END IF                ! END IF BLOCK
0526      IF(ISVP.NE.5) THEN
0527      CALL MOVE(200,399)      ! MOVE TO THESE COORDS
0528      CALL TEXT(9,'SSP AREA:') ! WRITE TEXT STRING
0529      CALL MOVE(272,399)      ! MOVE TO THESE COORDS
0530      CALL INUMBR(INDX,2)      ! WRITE NUMBER
0531      CALL MOVE(184,399)      ! MOVE TO THESE COORDS
0532      CALL TEXT(2,'--')      ! WRITE TEXT STRING
0533      END IF                ! END IF BLOCK
0534      CALL MOVE(360,397)      ! MOVE TO THESE COORDS
0535      CALL TEXT(4,'LAT:')      ! WRITE TEXT STRING
0536      CALL MOVE(400,397)      ! MOVE TO THESE COORDS
0537      CALL INUMBR(LAT(1),2)    ! WRITE NUMBER
0538      CALL MOVE(424,397)      ! MOVE TO THESE COORDS
0539      CALL INUMBR(LAT(2),2)    ! WRITE NUMBER
0540      CALL MOVE(448,397)      ! MOVE TO THESE COORDS
)41      CALL INUMBR(LAT(3),2)    ! WRITE NUMBER
0542      CALL MOVE(464,397)      ! MOVE TO THESE COORDS
0543      CALL TEXT(1,'-')      ! WRITE TEXT STRING
0544      CALL MOVE(472,397)      ! MOVE TO THESE COORDS
0545      CALL TEXT(1,LAT(4))      ! WRITE TEXT STRING
0546      CALL MOVE(504,397)      ! MOVE TO THESE COORDS
0547      CALL TEXT(5,'LONG:')    ! WRITE TEXT STRING
0548      CALL MOVE(552,397)      ! MOVE TO THESE COORDS
0549      CALL INUMBR(LONG(1),3)   ! WRITE NUMBER
0550      CALL MOVE(584,397)      ! MOVE TO THESE COORDS
0551      CALL INUMBR(LONG(2),2)    ! WRITE NUMBER
0552      CALL MOVE(608,397)      ! MOVE TO THESE COORDS
0553      CALL INUMBR(LONG(3),2)    ! WRITE NUMBER
0554      CALL MOVE(624,397)      ! MOVE TO THESE COORDS
0555      CALL TEXT(1,'-')      ! WRITE TEXT STRING
0556      CALL MOVE(632,397)      ! MOVE TO THESE COORDS
0557      CALL TEXT(1,LONG(4))      ! WRITE TEXT STRING
0558
0559      !-----LABEL X AXIS FOR BOTH GRAPHS
0560      CALL MOVE(88,371)      ! MOVE TO THESE COORDS
0561      CALL TEXT(18,'SOUND SPEED (FT/S)') ! WRITE TEXT STRING
0562      CALL MOVE(456,371)      ! MOVE TO THESE COORDS
0563      CALL TEXT(18,'SOUND SPEED (FT/S)') ! WRITE TEXT STRING
0564
0565      !-----LABEL Y AXIS FOR BOTH GRAPHS
0566      DO 359 J=0,376,376      ! DO AT 0 AND AT 376
0567      IYPOS=245              ! SET Y COORDINATE
)58      DO 358 I=1,8          ! DO EIGHT TIMES
0569      CALL MOVE(J,IYPOS)      ! MOVE TO THESE COORDS
0570      CALL TEXT(1,NL(I))      ! WRITE TEXT STRING

```

```

      1      IYPOS=IYPOS-14      ! RESET Y COORDINATE
0572      358      CONTINUE      ! END DO LOOP
0573      359      CONTINUE      ! END DO LOOP
0574
0575      !-----INSET XBT IN SHALLOW PROFILE
0576      IUD=(ISVP-1)*10+1      ! POINTER FOR JSPDAT ARRAY
0577      IF(ISVP.EQ.5) IUD=21      ! 'KEYPUNCHED' FLAG
0578      IDF=IUD+9      ! END POINTER
0579      IYPOS=217      ! SET Y COORDINATE
0580      DO 380 I=IUD,IDF      ! FOR PART OF JSPDAT ARRAY
0581          CALL MOVE(608,IYPOS)      ! MOVE TO THESE COORDS
0582          CALL TEXT(1,JSPDAT(I))      ! WRITE TEXT STRING
0583          IYPOS=IYPOS-14      ! RESET Y COORDINATE
0584      380      CONTINUE      ! END DO LOOP
0585      CALL MOVE(351,7)      ! MOVE TO THESE COORDS
0586      CALL LNTYPE(3)      ! TYPE OF LINE
0587      CALL DRAW(431,7)      ! DRAW TO THESE COORDS
0588      CALL LNTYPE(1)      ! TYPE OF LINE
0589      CALL DRAW(431,49)      ! DRAW TO THESE COORDS
0590      DO 390 NNN=7,49,14      ! DO AT 7,21,35,49
0591          CALL MOVE(431,NNN)      ! MOVE TO THESE COORDS
0592          CALL DRAW(435,NNN)      ! DRAW TO THESE COORDS
0593      390      CONTINUE      ! END DO LOOP
0594
0595      !-----PUT ON THE BOTTOM LINES OF TEXT
0596      CALL MOVE(72,35)      ! MOVE TO THESE COORDS
0597      CALL TEXT(30,'TRUE BOTTOM DEPTH      FATHOMS')! WRITE TEXT
0598      CALL MOVE(72,21)      ! MOVE TO THESE COORDS
0599      CALL TEXT(24,'TRUE WIND SPEED      KNOTS')! WRITE TEXT
0600      CALL MOVE(120,7)      ! MOVE TO THESE COORDS
0601      CALL TEXT(29,'***CANDIDATE ACOUSTIC PATHS>')! WRITE TEXT
0602      CALL MOVE(440,49)      ! MOVE TO THESE COORDS
0603      CALL TEXT(20,'LAYER DEPTH      FT')! WRITE TEXT STRING
0604      CALL MOVE(440,35)      ! MOVE TO THESE COORDS
0605      CALL TEXT(12,'SOUND CHNL.-')      ! WRITE TEXT STRING
0606      CALL MOVE(440,21)      ! MOVE TO THESE COORDS
0607      CALL TEXT(12,'CNVRG. ZONE.-')      ! WRITE TEXT STRING
0608      CALL MOVE(440,7)      ! MOVE TO THESE COORDS
0609      CALL TEXT(22,'BTM. BOUNCE-MGS PROV.')! WRITE TEXT STRING
0610      CALL MOVE(536,35)      ! MOVE TO THESE COORDS
0611      IF(SCHNLD.LE.SCHMXD) THEN      ! SOUND CHANNEL DEPTH
0612          CALL TEXT(13,'AXIS DEP=      ')! WRITE TEXT STRING
0613          CALL MOVE(608,35)      ! MOVE TO THESE COORDS
0614          IDEPSC=IIFIX(SCHNLD)      ! SOUND CHANNEL DEPTH
0615          CALL INUMBR(IDEPSC,3)      ! WRITE NUMBER
0616      ELSE      ! SOUND CHANNEL DEPTH>MAX
0617          CALL TEXT(10,'NOT USABLE')      ! WRITE TEXT STRING
0618      END IF      ! END IF BLOCK
0619      CALL MOVE(536,21)      ! MOVE TO THESE COORDS
0620      CALL TEXT(13,'RANGE      KYDS')      ! WRITE TEXT STRING
0621      CALL MOVE(584,21)      ! MOVE TO THESE COORDS
0622      IRCZ=IIFIX(RCZ)      ! RANGE OF CONVERG. ZONE
0623      CALL INUMBR(IRCZ,2)      ! WRITE NUMBER
0624      JBTMDP=IIFIX(BDF)      ! BOTTOM DEPTH IN FATHOMS
0625      CALL MOVE(216,35)      ! MOVE TO THESE COORDS
0626      CALL INUMBR(JBTMDP,4)      ! WRITE NUMBER
0627      JWIND=IIFIX(WS)      ! WIND SPEED

```

```

0628      CALL MOVE(200,21)           ! MOVE TO THESE COORDS
0629      CALL INUMBR(JWIND,2)        ! WRITE NUMBER
0630      LAYER=IIFIX(EL)             ! LAYER DEPTH
0631      CALL MOVE(528,49)           ! MOVE TO THESE COORDS
0632      CALL INUMBR(LAYER,5)        ! WRITE NUMBER
0633      CALL MOVE(624,7)            ! MOVE TO THESE COORDS
0634      CALL INUMBR(MGS,1)          ! WRITE NUMBER
0635      CALL MOVE(0,0)               ! DUMMY CALL TO DELAY EXIT
0636
0637      !-----HARDCOPY OPTION-----
0638      WRITE(5,800)                 ! PROMPT FOR HOW MANY COPIES
0639      READ(5,805) ICOPY             ! NUMBER OF COPIES WANTED
0640      IF(ICOPY.EQ.0) GOTO 900       ! NONE WANTED, SKIP NEXT
0641      DO 900 I = 1,ICOPY            ! DO FO NUMBER OF COPIES
0642          WRITE(5,850)              ! WRITE WHAT IS ON SCREEN
0643      900      CONTINUE             ! ! END DO LOOP
0644      CALL UNCLIP                   ! NO CLIPPING
0645      CALL ICLR                     ! CLEAR SCREEN
0646      999      RETURN              ! RETURN TO CALLING ROUTINE
0647
0648      !-----FORMAT STATEMENTS-----
0649      106      FORMAT(1H$, 'DO YOU WANT THE PROFILE DRAWN ON YOUR TEK-4025 (YES
0650      1 OR NO) ',T60,' ')
0651      191      FORMAT(' !BEL', '*** RAW DATA IN SHALLOW SVP WITH X''S ***',/)
0652      800      FORMAT(1H$, 'HOW MANY HARD COPIES WOULD YOU LIKE? [0,1,2,ETC.] ')
0653      805      FORMAT(I2)
0654      850      FORMAT(' !HCO S')
0655      950      FORMAT(A1)
0656
0657      END

```



```

0001          SUBROUTINE SVPRO(MONTH)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: SVPRO
0005      ! AUTHOR: S. KO, S. LAFLEUR & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1974, 1982 (REDESIGN) & 12/83 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE SVPRO WILL READ SOUND SPEED PROFILE ACCORDING
0008      !              TO SOUND SPEED INDEX OF THE OCEAN AREA. OTHER PARAMETERS
0009      !              WILL BE WILL ALSO BE DETERMINED FROM THE DATA BASE.
0010      ! INPUTS:  PARAMETERS PASSED IN. VARIABLES IN COMMONS.
0011      ! OUTPUTS: MODIFIED PARAMETERS PASSED OUT.
0012      ! MODULES CALLED: NONE
0013      ! CALLED BY: ENVIRN, FORCST
0014      !
0015      INCLUDE 'ENVN.INC'
0016 1 !-----ENVN-----
0017 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0018 1 ! -----
0019 1 ! BIO      (2)  BIOLOGICAL BACK SCATTERING  REAL*4  -57. & -47.
0020 1 ! DLYR                                REAL*4
0021 1 ! MGS                                INTEGER*2
0022 1
0023 1      REAL*4  BIO,DLYR
0024 1      INTEGER*2 MGS
0025 1      DATA BIO/-57.,-47./
0026 1
0027 1      COMMON /ENVN/ BIO(2),DLYR,MGS
0028 1
0029 1 !-----END ENVN-----
0030      INCLUDE 'GRF.INC'
0031 1 !-----GRF-----
0032 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0033 1 ! -----
0034 1 ! DBT      (25)  DEPTH OF DEPTH/VEL PAIR  REAL*4
0035 1 ! IANS                                INTEGER*2  -2 TO +2
0036 1 ! ILYR                                INTEGER*2
0037 1 ! INBT                                INTEGER*2
0038 1 ! ISVP                                INTEGER*2  1 OR 2
0039 1 ! I2000                                INTEGER*2
0040 1 ! VBT      (25)  VELOCITY FOR DEPTH PAIR  REAL*4  REAL*4
0041 1
0042 1      REAL*4  DBT,VBT
0043 1      INTEGER*2 IANS,ILYR,INBT,ISVP,I2000
0044 1
0045 1      COMMON /GRF/ IANS,ISVP,ILYR,I2000,INBT,DBT(25),VBT(25)
0046 1
0047 1 !-----END GRF-----
0048      INCLUDE 'LOC.INC'
0049 1 !-----LOC-----
0050 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0051 1 ! -----
0052 1 ! INDX                                INTEGER*2
0053 1 ! LAT      (4)  LATITUDE  INTEGER*2
0054 1 ! LONG      (4)  LONGITUDE  INTEGER*2
0055 1 ! NMAREA (20)  AREA OCEAN NAME  BYTE
0056 1 ! NOC                                INTEGER*2
0057 1 ! RCZ                                REAL*4
0058 1
0059 1      REAL*4  RCZ

```

```

0060 1      INTEGER*2 INDX,LAT, LONG, NOC
0061 1      BYTE      NMAREA(20)
0062 1
0063 1      COMMON /LOC/ LAT(4), LONG(4), NOC, INDX, RCZ, NMAREA
0064 1
0065 1 ! -----END LOC-----
0066      INCLUDE 'SVP.INC'
0067 1 ! -----SVP-----
0068 1 !  VARBL  SIZE      PURPOSE                                TYPE      RANGE
0069 1 !  -----
0070 1 !  BDF      BOTTOM DEPTH IN FATHOMS                      REAL*4
0071 1 !  BIOP     BIOLOGICAL BACK SCATTERING COEF             REAL*4
0072 1 !  BTDATE (9) DATE OF LAST BT INPUT                      BYTE
0073 1 !  BTTIME (8) TIME OF LAST BT INPUT                      BYTE
0074 1 !  C        (50) VELOCITY (PAIRED WITH Z FOR SVP)         REAL*4
0075 1 !  CC       (50) VELOCITY (PAIRED WITH ZZ FOR SVP)        REAL*4
0076 1 !  CS      SOUND VELOCITY AT SURFACE                     REAL*4
0077 1 !  DEG     TEMPERATURE (DEG)                             REAL*4      57.2957795
0078 1 !  EL      LAYER DEPTH                                    DATA
0079 1 !  F       FREQUENCY                                       REAL*4
0080 1 !  GRDS    GRIDS                                           REAL*4      0.0164
0081 1 !  ITO     MINIMAL 2-WAY TRAVEL TIME                     INTEGER*2
0082 1 !  MGSOP   MGS PROVINCE NUMBER                          INTEGER*2
0083 1 !  N       # OF DEPTH/VELOCITY PAIRS                     INTEGER*2
0084 1 !  NN      # OF DEPTH/VELOCITY PAIRS                     INTEGER*2
0085 1 !  PI      MATHEMATICAL CONSTANT PI                      REAL*4      3.1415927
0086 1 !  SNDATE (9) DATE SYS PARMS LAST UPDATED               BYTE
0087 1 !  SNTIME (8) TIME SYS PARMS LAST UPDTAED              BYTE
0088 1 !  SYDATE (9) CURRENT DATE READ FROM SYSTEM             BYTE
0089 1 !  SYTIME (8) CURRENT TIME READ FROM SYSTEM             BYTE
0090 1 !  TMP      TEMPERATURE                                    REAL*4
0091 1 !  UMKZ     BOTTOM BACK SCATTERING COEF.                  REAL*4      -28.0
0092 1 !  WS      WIND SPEED                                       REAL*4
0093 1 !  Z        (50) DEPTH OF POINT OF SOUND SPEED          REAL*4
0094 1 !  ZZ       (50) DEPTH OF POINT OF SOUND SPEED          REAL*4
0095 1
0096 1      INTEGER*2 ITO,MGSOP,N,NN
0097 1      REAL*4     BDF,BIOP,C(50),CC(50),CS,DEG,EL,F,GRDS
0098 1      REAL*4     PI,TMP,UMKZ,WS,Z(50),ZZ(50)
0099 1      BYTE      SYDATE(9),SYTIME(8),BTDATE(9),BTTIME(8)
0100 1      BYTE      SNDATE(9),SNTIME(8)
0101 1      DATA     PI,DEG,GRDS/3.1415927,57.2957795,0.0164/
0102 1      DATA     UMKZ/-28./
0103 1
0104 1      COMMON /SVP/ F,N,Z,C,EL,MGSOP,BDF,WS,CS,TMP,BIOP,
0105 1      1          UMKZ,PI,DEG,GRDS,ITO,ZZ,CC,NN,
0106 1      2          SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME
0107 1 ! -----SVP-END-----
0108      INCLUDE 'SVPl.INC'
0109 1 ! -----SVPl-----
0110 1 !  VARBL  SIZE      PURPOSE                                TYPE      RANGE
0111 1 !  -----
0112 1 !  BUFFER (224) HISTORICAL DATA FILE BUFFER             REAL*4
0113 1 !  DS      (30) HISTORICAL DEPTH                          REAL*4
0114 1 !  J20     # OF DEEP OCEAN DEPTH/VEL PAIRS                INTEGER*2
0115 1 !  NS      TOTAL # OF PAIRS IN HISTORICAL                 INTEGER*2
0116 1 !  NSN     MONTH NUMBER (1=JAN.,ETC)                     INTEGER*2      1 TO 12
0117 1 !  SLNTY   SALINITY                                       REAL*4
0118 1 !  VS      (30) HISTORICAL VELOCITY                      REAL*4

```

```

0119 1
0120 1 REAL*4 BUFFER,DS,SLNTY,VS
0121 1 INTEGER*2 J20,NSN,NS
0122 1
0123 1 COMMON /SVPL/ J20,BUFFER(224),NSN,SLNTY,DS(30),VS(30),NS
0124 1 !-----END SVPL-----
0125 !
0126 ! VARBL SIZE PURPOSE TYPE RANGE
0127 ! -----
0128 ! I COUNTER INTEGER*2
0129 ! ISN SEASON (INT DIVISION) INTEGER*2
0130 ! J COUNTER INTEGER*2
0131 ! JDV (5) NUMBER OF DEPTH/VEL PAIRS PER OCEAN INTEGER*2
0132 ! K COUNTER INTEGER*2
0133 ! LBIO POINTER INTEGER*2
0134 ! LNDX (5) LOGICAL RECORD LENGTH FOR OCEAN(1-5) INTEGER*2
0135 ! LOC POINTER INTEGER*2
0136 ! LREC LOGICAL RECORD LENGTH ARRAY INTEGER*2
0137 ! L0 POINTER INTEGER*2
0138 ! M NUMBER OF RECORDS IN FILE NOC INTEGER*2
0139 ! MM LENGTH OF RECORD (WORDS) IN NOC INTEGER*2
0140 ! MONTH NUMBER OF MONTH INTEGER*2 1 TO 12
0141 ! NAME (12) NAMES OF OCEAN AREA INTEGER*2
0142 ! OCNAME (3,5) NAMES OF OCEAN AREA INTEGER*2
0143 ! RECLEN (5) RECORD LENGTH INTEGER*2
0144 ! RECNUM (5) NUMBER OF RECORDS IN FILE INTEGER*2
0145 !
0146 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0147 !
0148 ! NOTE: THESE OCEANS HAVE 20 DEPTH/VEL PAIRS AND RECORD LENGTH 224:
0149 ! NORTH PACIFIC, NORTH ATLANTIC, INDIAN, NORWEGIAN.
0150 ! THE MEDITERRANEAN HAS 6 DEPTH/VEL PAIRS & RECORD LENGTH 196.
0151 !
0152 INTEGER*2 I,ISN,J,JDV,K,LBIO,LNDX,LOC,LREC,L0,M,MM,MONTH,NAME
0153 INTEGER*2 OCNAME,RECLEN,RECNUM
0154 DIMENSION JDV(5),LNDX(5),RECNUM(5),RECLEN(5),NAME(12)
0155 DIMENSION OCNAME(3,5)
0156
0157 DATA JDV/2*20,6,2*20/,LNDX/2*224,220,2*224/,
0158 1 RECNUM/69,77,5,50,45/,RECLEN/2*448,440,2*448/,
0159 2 NAME/' ',' ',' ',' ',' ',' ',' ',3*' ',' '.D','AT','','1',0/,
0160 3 OCNAME/'NP','AC',' ',' ','NL','AN','T','ME','D3','2','IN',
0161 4 'DI','AN','NR','W',' '/
0162
0163 !-----SET PARMS; OPEN SELECTED OCEAN
0164 ! ISN=(MONTH+2)/3 ! GET SEASON (INT DIVISION)
0165 ! L0=169 ! LOCATION FOR POINTER
0166 ! J20=JDV(NOC) ! SELECT D,V PAIRS
0167 ! IF(NOC.EQ.5.AND.INDX.LE.39) ! OCEAN=5 & INDEX<39
0168 1 J20=18 ! SET DEPTH/VEL PAIR POINTER
0169 ! LREC=LNDX(NOC) ! LOGICAL RECORD LENGTH
0170 ! NAME(6)=OCNAME(1,NOC) ! DATA FILE NAME 1ST 2 CHARS
0171 ! NAME(7)=OCNAME(2,NOC) ! SECOND 2 CHARS
0172 ! NAME(8)=OCNAME(3,NOC) ! THIRD 2 CHARS
0173 ! M=RECNUM(NOC) ! NUMBER OF RECORDS IN FILE
0174 ! MM=RECLEN(NOC) ! LENGTH OF RECORD (WORDS)
0175 CALL ASSIGN(3,NAME) ! ASSIGN FILE TO UNIT 3
0176 DEFINE FILE 3 (M,MM,U,INDX) ! FILE PARAMETERS
0177 READ(3'INDX) (BUFFER(I),I=1,LREC)! READ BUFFER

```

```

0178      INDX=INDX-1      ! DECREASE PROFILE INDEX
0179      CALL CLOSE(3)    ! CLOSE FILE 3
0180
0181      !-----SET MISC PARMS FROM HISTORICAL
0182      LOC=L0+2*(J20+1)  ! FACTOR FOR POINTER
0183      LBIO=LOC+2*(ISN-1) ! POINTER
0184      BIO(1)=BUFFER(LBIO) ! BIOLOGICAL BACKSCATTERING
0185      BIO(2)=BUFFER(LBIO+1) ! BIOLOGICAL BACKSCATTERING
0186      UMKZ=BUFFER(LOC+8) ! BOTTOM BACKSCATTERING
0187      SLNTY=BUFFER(LOC+9) ! SALINITY
0188
0189      !-----SET NEAR SURFACE DATA(4 TO 7 DA
0190      M=(MONTH-1)*14+1  ! MONTH POINTER
0191      DO 220 J=1,7      ! DO SEVEN TIMES
0192      DS(J)=BUFFER(M)   ! SET HISTORICAL DEPTH
0193      VS(J)=BUFFER(M+1) ! SET HISTORICAL VELOCITY
0194      IF(J.GT.1.AND.DS(J).LE.1.) GO TO 222 ! IF DEPTH < 1.0
0195      M=M+2             ! ADD 2 TO POINTER
0196      220 CONTINUE      ! END DO LOOP
0197      J=8               ! SET COUNTER TO EIGHT
0198      222 NS=J20+J-1    ! DEEP OCEAN UPPER LIMIT
0199
0200      !-----DEEP OCEAN STARTS AT ITEM 57 OF
0201      M=L0              ! L0 POINTS TO 57 OF BUFFER
0202      DO 225 K=J,NS     ! FOR DEEP OCEAN
0203      DS(K)=BUFFER(M)   ! SET HISTORICAL DEPTH
0204      VS(K)=BUFFER(M+1) ! SET HISTORICAL VELOCITY
0205      M=M+2             ! ADD 2 TO COUNTER
0206      225 CONTINUE      ! END DO LOOP
0207      DO 230 I=1,NS     ! FOR DEEP OCEAN
0208      IF(VS(I).EQ.0) GOTO 235 ! HISTORICAL VELOCITY = 0
0209      230 CONTINUE      ! END DO LOOP
0210      GO TO 999          ! RETURN TO CALLING ROUTINE
0211      235 NS=I-1        ! RESET UPPER LIMIT
0212      999 RETURN        ! RETURN TO CALLING ROUTINE
0213      END               ! END SUBROUTINE

```

```

0001      SUBROUTINE TEXT(LEN,STRNG)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: TEXT
0005      ! AUTHOR: J. CASCIO, W. WACHTER(FORTRAN 77), NUSC/NL, CODE 3333
0006      ! DATE: 1981 & 9/84 (FORTRAN 77)
0007      ! FUNCTION: WRITES OUT A STRING OF UP TO 80 CHARACTERS TO THE
0008      !              RIGHT OF THE CURRENT CURSOR POSITION WITH THE TERMINAL FONT.
0009      ! INPUTS: STRING TO BE WRITTEN
0010      ! OUTPUTS: WRITTEN STRING
0011      ! MODULES CALLED: NONE
0012      ! CALLED BY: INUMBR, SVPGRF
0013      !
0014      ! VARBL  SIZE PURPOSE                                TYPE      RANGE
0015      ! -----
0016      ! I          COUNTER                                INTEGER*2
0017      ! ILEN       LENGTH OF TEXT STRING                  INTEGER*2  0 TO 80
0018      ! STRNG      TEXT STRING TO BE WRITTEN              BYTE
0019      !
0020      BYTE STRNG(80)
0021      INTEGER*2 I,LEN
0022
0023      IF (LEN.GT.80 .OR. LEN.LT.0) LEN=80 ! RESET IF INVALID LENGTH
0024      TYPE 1,(STRNG(I),I=1,LEN)          ! WRITE STRING
0025      RETURN                              ! RETURN TO CALLING ROUTINE
0026
0027      !-----FORMAT STATEMENT-----
0028      1  FORMAT(' !STR "',<LEN>A1,'"')
0029      END

```

## SUBROUTINE VELTMP(DEPTH,SS,T1,SLNTY)

```

0001  SUBROUTINE VELTMP(DEPTH,SS,T1,SLNTY)
0002
0003  ! PROLOGUE:
0004  ! MODULE NAME: VELTMP
0005  ! AUTHOR: R. FLIGHT(VITRO) & W. WACHTER, CODE 3333, NUSC/NLL
0006  ! DATE: 1979 & 12/83 (FORTRAN 77)
0007  ! FUNCTION: SUBROUTINE VELTMP IS USED TO OBTAIN THE EQUIVALENT
0008  !             TEMPERATURE FOR A SPECIFIED SOUND SPEED AT A GIVEN
0009  !             DEPTH AND SALINITY.
0010  ! INPUTS:   PARAMETERS PASSED IN.
0011  ! OUTPUTS:  PARAMETERS PASSED OUT.
0012  ! MODULES CALLED: LEROY
0013  ! CALLED BY: ENVIRN, FORCST, METRIC
0014
0015  ! VARBL  SIZE      PURPOSE                      TYPE      RANGE
0016  ! -----
0017  ! DEPTH          DEPTH                      REAL*4
0018  ! NCK            COUNTER                    INTEGER*2
0019  ! SLNTY          SALINITY                   REAL*4
0020  ! SS            SOUND SPEED                 REAL*4
0021  ! T1            TEMPERATURE                 REAL*4
0022  ! V1            VELOCITY                   REAL*4
0023  !
0024  !             INTEGER*2 NCK
0025  !             REAL*4    DEPTH,SLNTY,SS,T1,V1
0026
0027  !             NCK=0
0028  !             T1=50.
0029  !             NCK=NCK+1
0030  !             CALL LEROY(DEPTH,T1,SLNTY,V1)
0031  !             V1=V1-SS
0032  !             IF (ABS(V1).GT..01.AND.NCK.LE.50) THEN
0033  !                 T1=T1-(V1*.16676)
0034  !                 GO TO 10
0035  !             ELSE
0036  !                 RETURN
0037  !             END IF
0038  !             END

```

10      INITIALIZE TEMP TO 50 DEGREES  
          INCREASE COUNTER  
          GET SOUND SPEED FOR TEMP  
          SOUND SPEED DIFFERENCE  
          SS DIFF > 0.01  
          3.29 M/SEC/DEGREE OR  
          .16676 IN ENGLISH UNITS  
          SS DIFF <= 0.01  
          RETURN TO CALLING ROUTINE  
          END IF BLOCK  
          END SUBROUTINE

```

0001      SUBROUTINE XBT(INSSP,NBT,NHIST,NEWBT)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: XBT
0005      ! AUTHOR: R. FLIGHT(VITRO) & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1979 & 12/83 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE XBT IS THE MAIN ROUTINE FOR THE XBT
0008      !               ERROR CORRECTING PROCESS.
0009      ! INPUTS: PARAMETERS PASSED IN AND VARIABLES IN COMMONS.
0010      ! OUTPUTS: NEW BT FLAG AND HISTORICAL DATA FLAG.
0011      ! MODULES CALLED: DUPDEP,DUPVEL,GLITCH,INSERT,LAYER,LEROY,
0012      !               LYRMOD,SMOOTH,SVPRO,XBTCHK,XBTERR,XBTGRF,XBTMOD
0013      ! CALLED BY: ENVIRN,FORCST
0014      !
0015      INCLUDE 'DTV.INC'
0016 1 ! -----DTV-----
0017 1 ! VARBL  SIZE  PURPOSE  TYPE  RANGE
0018 1 ! -----
0019 1 ! D      (25)  DEPTH      REAL*4
0020 1 ! DD     (25)  DEPTH      REAL*4
0021 1 ! NNBT   NUMBER OF BATHETHERMAL  INTEGER*2
0022 1 ! T      (25)  TEMPERATURE  REAL*4
0023 1 ! TT     (25)  TEMPERATURE  REAL*4
0024 1 ! VEL    (25)  VELOCITY    REAL*4
0025 1 !
0026 1      INTEGER*2 NNBT
0027 1      REAL*4 D,DD,T,TT,VEL
0028 1
0029 1      COMMON /DTV/ D(25),T(25),VEL(25),DD(25),TT(25),NNBT
0030 1 ! -----END DTV-----
0031      INCLUDE 'ENVN.INC'
0032 1 ! -----ENVN-----
0033 1 ! VARBL  SIZE  PURPOSE  TYPE  RANGE
0034 1 ! -----
0035 1 ! BIO    (2)   BIOLOGICAL BACK SCATTERING  REAL*4  -57. & -47.
0036 1 ! DLYR   LAYER DEPTH  REAL*4
0037 1 ! MGS    MGS PROVINCE  INTEGER*2
0038 1
0039 1      REAL*4 BIO,DLYR
0040 1      INTEGER*2 MGS
0041 1      DATA BIO/-57.,-47./
0042 1
0043 1      COMMON /ENVN/ BIO(2),DLYR,MGS
0044 1
0045 1 ! -----END ENVN-----
0046      INCLUDE 'GRF.INC'
0047 1 ! -----GRF-----
0048 1 ! VARBL  SIZE  PURPOSE  TYPE  RANGE
0049 1 ! -----
0050 1 ! DBT    (25)  DEPTH OF DEPTH/VEL PAIR  REAL*4
0051 1 ! IANS   PREDICTION TYPE  INTEGER*2  -2 TO +2
0052 1 ! ILYR   INDEX FOR LAYER DEPTH  INTEGER*2
0053 1 ! INBT   OPERATOR ENTERED # OF BT POINTS  INTEGER*2
0054 1 ! ISVP   LATEST OR HISTORICAL BT FLAG  INTEGER*2  1 OR 2
0055 1 ! I2000  SVP INDEX FOR 2000 FT DEPTH  INTEGER*2
0056 1 ! VBT    (25)  VELOCITY FOR DEPTH PAIR  REAL*4  REAL*4
0057 1

```

XBT

17-Dec-1984 12:43:41

17-Dec-1984 12:43:40

```

0058 1      REAL*4    DBT,VBT
0059 1      INTEGER*2 IANS,ILYR,INBT,ISVP,I2000
0060 1
0061 1      COMMON /GRF/ IANS,ISVP,ILYR,I2000,INBT,DBT(25),VBT(25)
0062 1
0063 1 ! -----END GRF-----
0064      INCLUDE 'LOC.INC'
0065 1 ! -----LOC-----
0066 1 !  VARBL  SIZE  PURPOSE                                TYPE  RANGE
0067 1 !  -----
0068 1 !  INDX      SSP INDEX                                INTEGER*2
0069 1 !  LAT      (4)  LATITUDE                                INTEGER*2
0070 1 !  LONG     (4)  LONGITUDE                                INTEGER*2
0071 1 !  NMAREA  (20)  AREA OCEAN NAME                        BYTE
0072 1 !  NOC      NUMBER OF OCEAN                            INTEGER*2
0073 1 !  RCZ      RANGE TO CONVERG. ZONE REAL*4
0074 1
0075 1      REAL*4    RCZ
0076 1      INTEGER*2 INDX,LAT,LONG,NOC
0077 1      BYTE      NMAREA(20)
0078 1
0079 1      COMMON /LOC/ LAT(4),LONG(4),NOC,INDX,RCZ,NMAREA
0080 1
0081 1 ! -----END LOC-----
0082      INCLUDE 'SVP.INC'
0083 1 ! -----SVP-----
0084 1 !  VARBL  SIZE  PURPOSE                                TYPE  RANGE
0085 1 !  -----
0086 1 !  BDF      BOTTOM DEPTH IN FATHOMS                        REAL*4
0087 1 !  BIOP      BIOLOGICAL BACK SCATTERING COEF            REAL*4
0088 1 !  BTDATE  (9)  DATE OF LAST BT INPUT                    BYTE
0089 1 !  BTTIME  (8)  TIME OF LAST BT INPUT                    BYTE
0090 1 !  C       (50)  VELOCITY (PAIRED WITH Z FOR SVP)        REAL*4
0091 1 !  CC      (50)  VELOCITY (PAIRED WITH ZZ FOR SVP)        REAL*4
0092 1 !  CS      SOUND VELOCITY AT SURFACE                      REAL*4
0093 1 !  DEG      TEMPERATURE (DEG)                            REAL*4      57.2957795
0094 1 !  EL      LAYER DEPTH                                    DATA
0095 1 !  F       FREQUENCY                                        REAL*4
0096 1 !  GRDS     GRIDS                                          REAL*4      0.0164
0097 1 !  ITO      MINIMAL 2-WAY TRAVEL TIME                    INTEGER*2
0098 1 !  MGSOP     MGS PROVINCE NUMBER                          INTEGER*2
0099 1 !  N       # OF DEPTH/VELOCITY PAIRS                      INTEGER*2
0100 1 !  NN      # OF DEPTH/VELOCITY PAIRS                      INTEGER*2
0101 1 !  PI      MATHEMATICAL CONSTANT PI                      REAL*4      3.1415927
0102 1 !  SNDATE  (9)  DATE SYS PARMS LAST UPDATED            BYTE
0103 1 !  SNTIME  (8)  TIME SYS PARMS LAST UPDTAED             BYTE
0104 1 !  SYDATE  (9)  CURRENT DATE READ FROM SYSTEM           BYTE
0105 1 !  SYTIME  (8)  CURRENT TIME READ FROM SYSTEM           BYTE
0106 1 !  TMP      TEMPERATURE                                    REAL*4
0107 1 !  UMKZ     BOTTOM BACK SCATTERING COEF.                  REAL*4      -28.0
0108 1 !  WS       WIND SPEED                                    REAL*4
0109 1 !  Z       (50)  DEPTH OF POINT OF SOUND SPEED           REAL*4
0110 1 !  ZZ      (50)  DEPTH OF POINT OF SOUND SPEED           REAL*4
0111 1
0112 1      INTEGER*2 ITO,MGSOP,N,NN
0113 1      REAL*4     BDF,BIOP,C(50),CC(50),CS,DEG,EL,F,GRDS
0114 1      REAL*4     PI,TMP,UMKZ,WS,Z(50),ZZ(50)

```



XBT

17-Dec-1984 12:43:41

17-Dec-1984 12:43:41

```

0115 1      BYTE      SYDATE(9),SYTIME(8),BTDATE(9),BTTIME(8)
0116 1      BYTE      SNDATE(9),SNTIME(8)
0117 1      DATA     PI,DEG,GRDS/3.1415927,57.2957795,0.0164/
0118 1      DATA     UMKZ/-28./
0119 1
0120 1      COMMON /SVP/ F,N,Z,C,EL,MGSOP,BDF,WS,CS,TMP,BIOP,
0121 1          1      UMKZ,PI,DEG,GRDS,ITO,ZZ,CC,NN,
0122 1          2      SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME
0123 1 ! -----SVP-END-----
0124      INCLUDE 'SVP1.INC'
0125 1 ! -----SVP1-----
0126 1 ! VARBL  SIZE      PURPOSE                                TYPE      RANGE
0127 1 ! -----
0128 1 ! BUFFER (224)    HISTORICAL DATA FILE BUFFER          REAL*4
0129 1 ! DS      (30)    HISTORICAL DEPTH                      REAL*4
0130 1 ! J20          # OF DEEP OCEAN DEPTH/VEL PAIRS          INTEGER*2
0131 1 ! NS          TOTAL # OF PAIRS IN HISTORICAL            INTEGER*2
0132 1 ! NSN         MONTH NUMBER (1=JAN.,ETC)                 INTEGER*2  1 TO 12
0133 1 ! SLNTY       SALINITY                                  REAL*4
0134 1 ! VS      (30)  HISTORICAL VELOCITY                     REAL*4
0135 1
0136 1      REAL*4      BUFFER,DS,SLNTY,VS
0137 1      INTEGER*2   J20,NSN,NS
0138 1
0139 1      COMMON /SVP1/ J20,BUFFER(224),NSN,SLNTY,DS(30),VS(30),NS
0140 1 ! -----END SVP1-----
0141 1
0142 1 ! VARBL  SIZE      PURPOSE                                TYPE      RANGE
0143 1 ! -----
0144 1 ! BAD          # OF POINTS NOT IN ACCEPTABLE RANGE        REAL*4
0145 1 ! CNT          TOTAL # OF POINTS LESS THAN 1500'          REAL*4
0146 1 ! ERRBT       ERROR FLAG FOR BT                          REAL*4
0147 1 ! HLYR       HISTORICAL LAYER DEPTH                      REAL*4
0148 1 ! I          COUNTER                                      INTEGER*2
0149 1 ! IMNTH      NUMBER OF MONTH                             INTEGER*2  1 TO 12
0150 1 ! INSSP     SOUND SPEED PROFILE INPUTTED                INTEGER*2
0151 1 ! J          COUNTER                                      INTEGER*2
0152 1 ! MONTH     NUMBER OF MONTH                             INTEGER*2  1 TO 12
0153 1 ! NBT       NUMBER OF BT POINTS                          INTEGER*2
0154 1 ! NDLYR     BT LAYER'S POSITION IN ARRAY                 INTEGER*2
0155 1 ! NEWBT     NEW BT FLAG                                  INTEGER*2
0156 1 ! NHIST     HISTORICAL DATA FLAG                        INTEGER*2
0157 1 ! NHLYR     HISTORICAL LAYER'S POSITION IN ARRAY          INTEGER*2
0158 1 ! NI        NUMBER OF HISTORIC DATA POINTS - 1          INTEGER*2
0159 1 ! N2        MERGE ROUTINE FLAG                            INTEGER*2
0160 1
0161 1 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0162 1
0163 1      INTEGER*2 I,IMNTH,INSSP,J
0164 1      INTEGER*2 MONTH,NBT,NDLYR,NEWBT,NHIST,NHLYR,NI,N2
0165 1      REAL*4      BAD,CNT
0166 1      REAL*4      ERRBT,HLYR,VEL1
0167 1
0168 1 ! -----PRELIMINARIES-----
0169 5      NHIST=0          ! HISTORICAL DATA FLAG
0170      NEWBT=0          ! NEW BT FLAG
( )1      N2=1           ! MERGE ROUTINE FLAG

```

XBT

17-Dec-1984 12:43:41

17-Dec-1984 12:43:40

```

0172      CNT=0.                ! COUNT OF # OF BT POINTS
0173      IMNTH=0              ! MONTH NUMBER
0174      INBT=NBT             ! # OF BT POINTS
0175      ERRET=0.            ! BT ERROR FLAG
0176      BAD=0.              ! # OF INVALID POINTS
0177      MONTH = NSN          ! MONTH NUMBER
0178      DO 20 I=1,NBT        ! DO FOR NUMBER OF BT
0179          IF(D(I).GT.2500.) GO TO 40 ! >2500 FEET, SKIP NEXT
0180          IF(D(I).LT.1500.) CNT=CNT+1. ! CNT= # OF BT PTS <1500
0181          CALL LEROY(D(I),T(I),SLNTY,VBT(I)) ! CONVERT FROM DEPTH/TEMP
0182          DBT(I)=D(I)       ! STORE DEPTH/VEL PAIRS
0183          VEL(I)=VBT(I)     ! STORE DEPTH/VEL PAIRS
0184      20  CONTINUE          ! END DO LOOP
0185      40  CALL XBTGRF(MONTH) ! GRAPH XBT
0186          CALL SVPRO(MONTH)  ! GET HIST DATA
0187          CALL LAYER(NS,DS,VS,HLYR) ! GET HIST LAYER DEPTH
0188          CALL INSERT(NS,DS,VS,HLYR,NHLYR) ! HIST LAYER'S POSITION
0189          CALL LAYER(NBT,D,VEL,DLYR) ! GET BT LAYER DEPTH
0190          CALL INSERT(NBT,D,VEL,DLYR,NDLYR) ! GET BT LAYER'S POSITION
0191      50  DO 50 J=1,NBT      ! DO FOR # OF BT POINTS
0192          IF(T(J).LT.27..OR.T(J).GT.95.) THEN ! OUTSIDE TEMP RANGE
0193              IF(D(J).LT.1500.) BAD=BAD+1. ! INVALID IF < 1500'
0194              IF(BAD.GT.(CNT*.5)) THEN ! > HALF ARE INVALID,
0195                  ERRET=1. ! SET ERROR FLAG
0196                  CALL XBTErr(INSSP,NBT,ERRET,NHIST,NEWBT,NDLYR) ! CORRECT BT
0197                  GO TO 999 ! BACK TO CALLING ROUTINE
0198                  END IF ! END IF BLOCK
0199                  END IF ! END IF BLOCK
0200      50  CONTINUE          ! END DO LOOP
0201          CALL DUPDEF(NBT,D,VEL) ! RID DOUBLE CONSEC DEPS
0202          CALL DUPVEL(NBT,D,VEL) ! RID DOUBLE CONSEC VELS
0203          NI=NS-1           ! # OF HIST DATA POINTS-1
0204          CALL XBTCHK(NBT,CNT,BAD,NI) ! CHECK BT DATA
0205          IF(BAD.GT.CNT*.5) THEN ! MORE THAN HALF ARE BAD
0206              ERRET=2. ! SET BT ERROR FLAG
0207              CALL XBTErr(INSSP,NBT,ERRET,NHIST,NEWBT,NDLYR) ! CORRECT
0208              IF(NEWBT.EQ.2) THEN ! NEW SSP AREA CHOSEN
0209                  CALL SVPRO(MONTH) ! GET HIST DATA
0210                  GOTO 5 ! START AGAIN
0211                  END IF ! END IF BLOCK
0212                  GO TO 999 ! BACK TO CALLING ROUTINE
0213                  END IF ! END IF BLOCK
0214
0215      !-----MODIFY LAYER DEPTH IF REQUIRED
0216          CALL LYRMOD(NBT,DLYR,NDLYR,HLYR) ! FORCE BT LAYER DEPTH
0217
0218      !-----IF XBT POINTS ARE OUTSIDE TOLERANCE ENVELOPE, MOVE THEM TO ENVELOPE
0219          CALL XBTMOD(NBT,NI,N2,NDLYR) ! MODIFY XBT IF REQUIRED
0220          DLYR=D(NDLYR) ! LAYER DEPTH
0221
0222      !-----REMOVE GLITCHES BELOW LAYER
0223          IF(NBT.GE.NDLYR+3) CALL GLITCH(NBT,NDLYR)
0224          CALL DUPDEF(NBT,D,VEL) ! RID DOUBLE CONSEC DEPS
0225          CALL DUPVEL(NBT,D,VEL) ! RID DOUBLE CONSEC VELS
0226          CALL LAYER(NBT,D,VEL,DLYR) ! GET BT LAYER DEPTH
0227          CALL INSERT(NBT,D,VEL,DLYR,NDLYR) ! GET LAYER'S POSITION
0228

```

XBT

17-Dec-1984 12:43:41

17-Dec-1984 12:43:40

```
0229  !-----SMOOTH XBT DATA-----
0230      IF(BAD.NE.0..AND.NBT.GE.3) CALL SMOOTH(NBT,NDLYR)
    B1      N2=0      ! INITIALIZE FLAG
0232      CALL XBTMOD(NBT,NI,N2,NDLYR)      ! CHECK BT VALUES VS HIST
0233      999      RETURN      ! BACK TO CALLING ROUTINE
0234      END
```

#### COMMAND QUALIFIERS

FORTRAN /CHECK=ALL/LIST/SHOW=(INCLUDE,NOMAP) ILAFLEURIXBT.F77

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE\_FORM)

/SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)

/F77 /NOG\_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD\_LINES /NOCROSS\_REFERENCE /I

#### COMPILATION STATISTICS

Run Time: 2.57 seconds

Elapsed Time: 5.38 seconds

Page Faults: 380

Dynamic Memory: 147 pages

```

0001      SUBROUTINE XBTCHK(NBT,CNT,BAD,NI)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: XBTCHK
0005      ! AUTHOR: S. LAFLEUR, W. WACHTER (FORTRAN 77)
0006      ! DATE: 7/84 & 7/84 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE XBTCHK DETERMINES ACCEPTANCE OR REJECTION OF
0008      !              THE BT DATA. IT INTERPOLATES THE SOUND SPEEDS OF THE BT
0009      !              AND HISTORICAL DATA AT EVERY 4 FEET OF DEPTH FROM SURFACE
0010      !              TO 1500 FEET OR THE LAST DEPTH ENTERED FOR THE BT IF IT IS
0011      !              LESS THAN 1500 FEET. 'CNT' IS INCREMENTED AT EVERY ONE
0012      !              OF THESE DEPTHS. THE TOLERANCE ENVELOPE IS ALSO
0013      !              CALCULATED AT EVERY ONE OF THESE DEPTHS. 'BAD' IS
0014      !              INCREMENTED EVERY TIME THE BT SOUND SPEED VALUE LIES
0015      !              OUTSIDE THE TOLERANCE ENVELOPE ABOUT THE HISTORICAL SOUND
0016      !              SPEED AT THE CURRENT DEPTH. IF MORE THAN HALF OF
0017      !              THE 'CNT' DEPTHS ARE 'BAD', THE BT WILL BE REJECTED.
0018      ! INPUTS: PARAMETERS PASSED IN & VARIABLES IN COMMONS.
0019      ! OUTPUTS: MODIFY SS TO STAY WITHIN TOLERANCE ENVELOPE
0020      ! MODULES CALLED: NONE
0021      ! CALLED BY: XBT
0022      !
0023      INCLUDE 'DTV.INC'
0024      1 !-----DTV-----
0025      1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0026      1 ! -----
0027      1 ! D      (25)  DEPTH                                REAL*4
0028      1 ! DD     (25)  DEPTH                                REAL*4
0029      1 ! NNBT                                INTEGER*2
0030      1 ! T      (25)  TEMPERATURE                          REAL*4
0031      1 ! TT     (25)  TEMPERATURE                          REAL*4
0032      1 ! VEL    (25)  VELOCITY                             REAL*4
0033      1 !
0034      1      INTEGER*2 NNBT
0035      1      REAL*4 D,DD,T,TT,VEL
0036      1
0037      1      COMMON /DTV/ D(25),T(25),VEL(25),DD(25),TT(25),NNBT
0038      1 !-----END DTV-----
0039      INCLUDE 'SVP1.INC'
0040      1 !-----SVP1-----
0041      1 ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0042      1 ! -----
0043      1 ! BUFFER (224) HISTORICAL DATA FILE BUFFER        REAL*4
0044      1 ! DS      (30) HISTORICAL DEPTH                    REAL*4
0045      1 ! J20                                INTEGER*2
0046      1 ! NS      TOTAL # OF PAIRS IN HISTORICAL            INTEGER*2
0047      1 ! NSN     MONTH NUMBER (1=JAN.,ETC)                INTEGER*2  1 TO 12
0048      1 ! SLNTY                                REAL*4
0049      1 ! VS      (30) HISTORICAL VELOCITY                 REAL*4
0050      1
0051      1      REAL*4 BUFFER,DS,SLNTY,VS
0052      1      INTEGER*2 J20,NSN,NS
0053      1
0054      1      COMMON /SVP1/ J20,BUFFER(224),NSN,SLNTY,DS(30),VS(30),NS
0055      1 !-----END SVP1-----
0056      !
0057      ! VARBL  SIZE  PURPOSE                                TYPE      RANGE
0058      ! -----
0059      ! BAD                                POINTS OUTSIDE TOLERANCE ENVELOPE REAL*4

```

```

0060      ! BTSPD          BT SS AT CURRENT DEPTH          REAL*4
0061      ! CNT           NUMBER OF POINTS <= 1500 FEET    REAL*4
0062      ! DEP           DEPTH                             REAL*4
0063      ! HSPD          HISTORICAL SS AT CURRENT DEPTH    REAL*4
0064      ! IDEP          DEPTH                             INTEGER*2
0065      ! K             LOOP COUNTER                     INTEGER*2
0066      ! NBT           NUMBER OF BT POINTS               INTEGER*2
0067      ! NI            # OF HISTORICAL POINTS            INTEGER*2
0068      ! XX            1/2 ENVELOPE TOLERANCE WIDTH      REAL*4
0069      !              AT CURRENT DEPTH
0070      !
0071      ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0072
0073      INTEGER*2 IDEP,K,NBT,NI
0074      REAL*4     BAD,BTSPD,CNT,DEP,HSPD,XX
0075
0076      !-----PRELIMINARIES-----
0077      CNT=0.          ! INIT # OF PTS <= 1500'
0078      BAD=0.          ! INIT # OF PTS OUTSIDE ENVELOPE
0079      DO 100 IDEP=0,1500,4      ! TO TO 1500' BY 4
0080          DEP=FLOAT(IDEP)      ! REAL NUMBER DEPTH
0081          IF(DEP.GT.D(NBT)) GO TO 999 ! TOO DEEP, RETURN TO CALLING RO
0082          IF(DEP.LE.1500.) CNT=CNT+1. ! DEPTH<=1500', INCREASE COUNT
0083
0084      !-----GET BT SS AT CURRENT DEPTH-----
0085          DO 200 K=1,NBT-1      ! DO UNTIL NEXT TO LAST BT
0086              IF(D(K).LE.DEP.AND.D(K+1).GT.DEP) GOTO 300 ! EXIT LOOP
0087      200      CONTINUE          ! END DO LOOP
0088      300      BTSPD=VEL(K)+((VEL(K+1)-VEL(K))/(D(K+1)-D(K)))*(DEP-D(K)) ! BT
0089
0090      !-----GET HISTORIC SS AT CURRENT DEPTH-----
0091          DO 500 K=1,NI          ! COMPARE DEPTHS LOOP
0092              IF(DS(K).LE.DEP.AND.DS(K+1).GT.DEP) GOTO 600 ! EXIT LOOP
0093      500      CONTINUE          ! END DO LOOP
0094      600      HSPD=VS(K)+((VS(K+1)-VS(K))/(DS(K+1)-DS(K)))*(DEP-DS(K)) ! HIS
0095
0096      !-----ENVELOPE CHECK-----
0097          XX=15.-.006*DEP      ! 1/2 ENVELOPE TOLERANCE WIDTH
0098          IF(DEP.LE.1500.AND.HSPD+XX.LT.BTSPD) BAD=BAD+1. ! ENVELOPE CHE
0099          IF(DEP.LE.1500.AND.HSPD-XX.GT.BTSPD) BAD=BAD+1. ! ENVELOPE CHE
0100      100      CONTINUE          ! END DO LOOP
0101      999      RETURN            ! RETURN TO CALLING ROUTINE
0102      END                        ! END SUBROUTINE

```

## SUBROUTINE XBTErr(INSSP,NBT,ERRBT,NHIST,NEWBT,NDLYR)

```

)1
0002
0003 ! PROLOGUE:
0004 ! MODULE NAME: XBTErr
0005 ! AUTHOR: R. FLIGHT(VITRO) & W. WACHTER, CODE 3333, NUSC/NLL
0006 ! DATE: 1979 & 12/83 (FORTRAN 77)
0007 ! FUNCTION: SUBROUTINE XBTErr PRODUCES THE ERROR MESSAGES
0008 !           AND ALLOWS THE OPERATOR TO SELECT EITHER NEW BT,
0009 !           HISTORICAL DATA, OR ADJUSTED BT DATA.
0010 ! INPUTS: OPERATOR SELECTION FOR BT DATA.  PARAMETERS PASSED IN.
0011 !         VARIABLES IN COMMONS.
0012 ! OUTPUTS: CRT PROMPTING MESSAGES TO OPERATOR
0013 ! MODULES CALLED: ASIS, INSERT, METRIC
0014 ! CALLED BY: XBT
0015 !
0016 !           INCLUDE 'DTV.INC'
0017 1 ! -----DTV-----
0018 1 ! VARBL  SIZE  PURPOSE  TYPE  RANGE
0019 1 ! -----
0020 1 ! D      (25)  DEPTH      REAL*4
0021 1 ! DD     (25)  DEPTH      REAL*4
0022 1 ! NNBT   NUMBER OF BATHETHERMAL  INTEGER*2
0023 1 ! T      (25)  TEMPERATURE  REAL*4
0024 1 ! TT     (25)  TEMPERATURE  REAL*4
0025 1 ! VEL    (25)  VELOCITY    REAL*4
0026 1 !
0027 1 !           INTEGER*2 NNBT
0028 1 !           REAL*4 D,DD,T,TT,VEL
0029 1 !
0030 1 !           COMMON /DTV/  D(25),T(25),VEL(25),DD(25),TT(25),NNBT
0031 1 ! -----END DTV-----
0032 !           INCLUDE 'ENVN.INC'
0033 1 ! -----ENVN-----
0034 1 ! VARBL  SIZE  PURPOSE  TYPE  RANGE
0035 1 ! -----
0036 1 ! BIO    (2)   BIOLOGICAL BACK SCATTERING  REAL*4  -57. & -47.
0037 1 ! DLYR   LAYER DEPTH  REAL*4
0038 1 ! MGS    MGS PROVINCE  INTEGER*2
0039 1 !
0040 1 !           REAL*4  BIO,DLYR
0041 1 !           INTEGER*2 MGS
0042 1 !           DATA BIO/-57.,-47./
0043 1 !
0044 1 !           COMMON /ENVN/  BIO(2),DLYR,MGS
0045 1 !
0046 1 ! -----END ENVN-----
0047 !           INCLUDE 'GRF.INC'
0048 1 ! -----GRF-----
0049 1 ! VARBL  SIZE  PURPOSE  TYPE  RANGE
0050 1 ! -----
0051 1 ! DBT    (25)  DEPTH OF DEPTH/VEL PAIR  REAL*4
0052 1 ! IANS   PREDICTION TYPE  INTEGER*2  -2 TO +2
0053 1 ! ILYR   INDEX FOR LAYER DEPTH  INTEGER*2
0054 1 ! INBT   OPERATOR ENTERED # OF BT POINTS  INTEGER*2
0055 1 ! ISVP   LATEST OR HISTORICAL BT FLAG  INTEGER*2  1 OR 2
0056 1 ! IZ000  SVP INDEX FOR 2000 FT DEPTH  INTEGER*2
0057 1 ! VBT    (25)  VELOCITY FOR DEPTH PAIR  REAL*4

```

XBTRR

17-Dec-1984 15:41:30

17-Dec-1984 15:41:28

```

0058 1
0059 1      REAL*4    DMT,VBT
0060 1      INTEGER*2 IANS,ILYR,INBT,ISVP,I2000
0061 1
0062 1      COMMON /GRF/ IANS,ISVP,ILYR,I2000,INBT,DBT(25),VBT(25)
0063 1
0064 1 ! -----END GRF-----
0065      INCLUDE 'LOC.INC'
0066 1 ! -----LOC-----
0067 1 !  VARBL  SIZE  PURPOSE  TYPE  RANGE
0068 1 !  -----
0069 1 !  INDX      SSP INDEX  INTEGER*2
0070 1 !  LAT      (4)  LATITUDE  INTEGER*2
0071 1 !  LONG     (4)  LONGITUDE  INTEGER*2
0072 1 !  NMAREA  (20)  AREA OCEAN NAME  BYTE
0073 1 !  NOC      NUMBER OF OCEAN  INTEGER*2
0074 1 !  RCZ      RANGE TO CONVERG. ZONE REAL*4
0075 1
0076 1      REAL*4    RCZ
0077 1      INTEGER*2 INDX,LAT,LONG,NOC
0078 1      BYTE      NMAREA(20)
0079 1
0080 1      COMMON /LOC/ LAT(4),LONG(4),NOC,INDX,RCZ,NMAREA
0081 1
0082 1 ! -----END LOC-----
0083      INCLUDE 'SVP.INC'
0084 1 ! -----SVP-----
0085 1 !  VARBL  SIZE  PURPOSE  TYPE  RANGE
0086 1 !  -----
0087 1 !  BDF      BOTTOM DEPTH IN FATHOMS  REAL*4
0088 1 !  BIOP     BIOLOGICAL BACK SCATTERING COEF  REAL*4
0089 1 !  BTDATE  (9)  DATE OF LAST BT INPUT  BYTE
0090 1 !  BTIME   (8)  TIME OF LAST BT INPUT  BYTE
0091 1 !  C        (50)  VELOCITY (PAIRED WITH Z FOR SVP)  REAL*4
0092 1 !  CC       (50)  VELOCITY (PAIRED WITH ZZ FOR SVP)  REAL*4
0093 1 !  CS      SOUND VELOCITY AT SURFACE  REAL*4
0094 1 !  DEG     TEMPERATURE (DEG)  REAL*4  57.2957795
0095 1 !  EL      LAYER DEPTH  DATA
0096 1 !  F        FREQUENCY  REAL*4
0097 1 !  GRDS     GRIDS  REAL*4  0.0164
0098 1 !  ITO      MINIMAL 2-WAY TRAVEL TIME  INTEGER*2
0099 1 !  MGSOP    MGS PROVINCE NUMBER  INTEGER*2
0100 1 !  N        # OF DEPTH/VELOCITY PAIRS  INTEGER*2
0101 1 !  NN       # OF DEPTH/VELOCITY PAIRS  INTEGER*2
0102 1 !  PI      MATHEMATICAL CONSTANT PI  REAL*4  3.1415927
0103 1 !  SNDATE  (9)  DATE SYS PARMS LAST UPDATED  BYTE
0104 1 !  SNTIME  (8)  TIME SYS PARMS LAST UPDTAED  BYTE
0105 1 !  SYDATE  (9)  CURRENT DATE READ FROM SYSTEM  BYTE
0106 1 !  SYTIME  (8)  CURRENT TIME READ FROM SYSTEM  BYTE
0107 1 !  TMP      TEMPERATURE  REAL*4
0108 1 !  UMKZ     BOTTOM BACK SCATTERING COEF.  REAL*4  -28.0
0109 1 !  WS       WIND SPEED  REAL*4
0110 1 !  Z        (50)  DEPTH OF POINT OF SOUND SPEED  REAL*4
0111 1 !  ZZ       (50)  DEPTH OF POINT OF SOUND SPEED  REAL*4
0112 1
0113 1      INTEGER*2 ITO,MGSOP,N,NN
0114 1      REAL*4    BDF,BIOP,C(50),CC(50),CS,DEG,EL,F,GRDS

```

C-54.2

XBTErr

17-Dec-1984 15:41:30

17-Dec-1984 15:41:28

```

C 15 1      REAL*4      PI,TMP,UMKZ,WS,Z(50),ZZ(50)
C 16 1      BYTE       SYDATE(9),SYTIME(8),BTDATE(9),BTTIME(8)
0117 1      BYTE       SNDATE(9),SNTIME(8)
0118 1      DATA      PI,DEG,GRDS/3.1415927,57.2957795,0.0164/
0119 1      DATA      UMKZ/-28./
0120 1
0121 1      COMMON /SVP/ F,N,Z,C,EL,MGSOP,BDF,WS,CS,TMP,BIOP,
0122 1          1      UMKZ,PI,DEG,GRDS,ITD,ZZ,CC,NN,
0123 1          2      SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME
0124 1 ! -----SVP-END-----
0125      INCLUDE 'SVP1.INC'
0126 1 ! -----SVP1-----
0127 1 !  VARBL  SIZE      PURPOSE                                TYPE      RANGE
0128 1 !  -----
0129 1 !  BUFFER (224)    HISTORICAL DATA FILE BUFFER          REAL*4
0130 1 !  DS      (30)    HISTORICAL DEPTH                      REAL*4
0131 1 !  J20                                # OF DEEP OCEAN DEPTH/VEL PAIRS INTEGER*2
0132 1 !  NS      TOTAL # OF PAIRS IN HISTORICAL              INTEGER*2
0133 1 !  NSN      MONTH NUMBER (1=JAN.,ETC)                   INTEGER*2  1 TO 12
0134 1 !  SLNTY      SALINITY                                  REAL*4
0135 1 !  VS      (30)    HISTORICAL VELOCITY                  REAL*4
0136 1
0137 1      REAL*4      BUFFER,DS,SLNTY,VS
0138 1      INTEGER*2   J20,NSN,NS
0139 1
0140 1      COMMON /SVP1/ J20,BUFFER(224),NSN,SLNTY,DS(30),VS(30),NS
0141 1 ! -----END SVP1-----
0142 1
C 143 1 !  VARBL  SIZE      PURPOSE                                TYPE      RANGE
0144 1 !  -----
0145 1 !  ERRBT      BT ERROR FLAG                                REAL*4
0146 1 !  HLYR      HISTORICAL LAYER DEPTH                      REAL*4
0147 1 !  I        LOOP COUNTER                                INTEGER*2
0148 1 !  IERROR     METRIC ERROR FLAG                          INTEGER*2
0149 1 !  INDXH      INDEX OF POINT INSERTED IN HIST             INTEGER*2
0150 1 !  INSSP      INPUT SSP                                  INTEGER*2
0151 1 !  K        COUNTER                                    INTEGER*2
0152 1 !  KUP       MAX OF SVP AT 1500' AND INPUTTED #          INTEGER*2
0153 1 !  NBT       NUMBER OF BT POINTS                          INTEGER*2
0154 1 !  NDLYR      INDEX OF BT LAYER DEPTH                      INTEGER*2
0155 1 !  NEWBT      OPERATOR RESPONSE TO NEW BT PROMPT          INTEGER*2
0156 1 !  NHIST      HISTORICAL DATA FLAG                       INTEGER*2
0157 1 !  N15       HISTORICAL SVP INDEX AT 1500'                INTEGER*2
0158 1 !  SPDDIF     SOUND SPEED DIFFERENCE (BT VS HIST)        REAL*4
0159 1
0160 1 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMON ***
0161
0162      INTEGER*2 1,IERROR,INDXH,INSSP,K,KUP,NBT,NDLYR,NEWBT,NHIST,N15
0163      REAL*4     ERRBT,HLYR,SPDDIF
0164
0165 1 ! -----PROMPT OPERATOR-----
0166      IF(ERRBT.NE.2.) THEN      ! ERROR FLAG NOT 2
0167          WRITE (5,272)          ! WRITE ERROR MESSAGE AND
0168      ELSE                      ! ERROR FLAG IS 2
0169          WRITE(5,276)           ! WRITE ERROR MESSAGE
0170      END IF                    ! END IF BLOCK
0171      WRITE(5,290)              ! WRITE HISTORICAL DATA

```



```

0172      CALL INSERT(NS,DS,VS,1500.,N15)! INSERT POINT IN SVP
0173      KUP=MAX0(N15,INBT)           ! MAX OF SVP INDEX AND IN BT
0174      DO 315 K=1,KUP              ! MAX OF SVP INDEX AND IN BT
0175          IF(K.GT.INBT.OR.K.GT.N15) GO TO 318 ! SKIP NEXT
0176          IF(K.GT.INBT.AND.K.LE.N15)         ! > INPUTTED #, <= 1500' SVP
0177      1      WRITE(5,310) K,DS(K),VS(K) ! WRITE DATA
0178          IF(K.LE.INBT.AND.K.GT.N15)         ! <= INPUTTED #, > 1500' SVP
0179      1      WRITE(5,312) K,DBT(K),VBT(K)!WRITE DATA
0180          IF(K.LE.INBT.AND.K.LE.N15)         ! <= INPUTTED # AND 1500' SVP
0181      1      WRITE(5,310) K,DS(K),VS(K),K,DBT(K),VBT(K) ! WRITE DATA
0182      315      CONTINUE                ! END DO LOOP
0183          IF(ERRBT.NE.2.) THEN          ! ERROR FLAG NOT 2
0184              WRITE(5,316)              ! PROMPT NEW BT OR HISTORICAL
0185          ELSE                          ! ERROR FLAG IS 2
0186      318      WRITE(5,320)              ! PROMPT NEW BT OR ADJUSTED DATA
0187              WRITE(5,321)              ! PROMPT USE BT AS IS
0188          END IF                        ! END IF BLOCK
0189      325      READ (5,330) NEWBT        ! OPERATOR RESPONSE
0190          IF(ERRBT.EQ.1..AND.NEWBT.EQ.2) NHIST=1 ! USE HISTORICAL SSP
0191          IF(ERRBT.EQ.2..AND.NEWBT.EQ.3) NHIST=1 ! USE HISTORICAL SSP
0192          IF(ERRBT.EQ.1..OR.NEWBT.EQ.1.OR.NEWBT.EQ.3) GO TO 400 ! GO TO EXIT
0193
0194      !-----OPERATOR WANTS NEW SSP AREA-----
0195          IF(NEWBT.EQ.2) THEN            ! NEW SSP AREA
0196              CALL ICLK                  ! CLEAR SCREEN
0197              WRITE(5,340)              ! PROMPT FOR NEW SSP AREA
0198              READ(5,330) INDX          ! NEW BT AREA
0199              NBT=NNBT                  ! NUMBER OF BT
0200              DO 342 I=1,NBT            ! DO FOR NUMBER OF BT
0201                  D(I)=DD(I)            ! STORE DEPTH
0202                  T(I)=TT(I)            ! STORE TEMPERATURE
0203      342      CONTINUE                ! END DO LOOP
0204              CALL METRIC(INSSP,D,T,NBT,Z,C,SLNTY,VS(1),IERROR) ! METRIC
0205              GO TO 400                  ! GO TO RETURN
0206          END IF                        ! END IF BLOCK
0207
0208      !-----CHECK METHOD OF CORRECTION-----
0209          IF(NEWBT.EQ.4) THEN            ! FORCE BT TO FIT HISTORICAL
0210              IF(NDLYR.NE.NBT.AND.NDLYR.NE.1) THEN ! IF LAYER <> 1ST OR LAST
0211                  INBT=NDLYR+1          ! RESET NUMBER OF POINTS IN BT
0212                  NBT=INBT              ! RESET NUMBER OF POINTS IN BT
0213              END IF                    ! END IF BLOCK
0214              CALL INSERT(NS,DS,VS,DBT(NBT),INDXH) ! INSERT PT AT LAST BT PT
0215              SPDDIF=VBT(NBT)-VS(INDXH) ! SOUND SPEED DIFFERENCE
0216              DO 350 I=1,NBT            ! DO FOR NUMBER OF BT
0217                  VBT(I)=VBT(I)-SPDDIF ! SUBTRACT SS DIFFERENCE
0218      350      CONTINUE                ! END DO LOOP
0219          END IF                        ! END IF BLOCK
0220          CALL ASIS(INBT,DBT,VBT)        ! USE BT AS IS
0221      400      RETURN                  ! RETURN TO CALLING ROUTINE
0222
0223      !-----FORMAT STATEMENTS-----
0224      272      FORMAT(/1H0,15X,'PROBABLE ERROR IN XBT'
0225              1/1H ,10X,'NEAR SURFACE WIRE BREAK USE HISTORICAL DATA')
0226      276      FORMAT(/1H0,10X,'PROBABLE ERROR IN XBT')
0227      290      FORMAT(/1H0,10X,'HISTORICAL DATA',T52,'XBT DATA'
0228              1/1H ,T11,'DEPTH',T21,'VEL',T50,'DEPTH',T60,'VEL'//)

```

```

0229      310      FORMAT(I5,2F10.1,T40,I5,2F10.1)
      B0      312      FORMAT(T40,I5,2F10.1)
0231      316      FORMAT(/1H0,T30,'RECOMMEND NEW XBT BE TAKEN'
0232      1/1H ,T25,'IF NOT HISTORICAL DATA WILL BE USED'
0233      2/1H ,T25,'NEAR SURFACE WIRE BREAK WILL NOT ALLOW'
0234      3/1H ,T25,'DATA TO BE ADJUSTED'
0235      4/1H ,T25,'INDICATE YOUR CHOICE'
0236      5/1H ,T25,'1=NEW XBT OR EDIT CURRENT XBT'
0237      6/1H$,T25,'2=USE HISTORICAL SSP',T60)
0238      320      FORMAT(/1H0,T20,'1=NEW XBT OR EDIT CURRENT XBT',
0239      1      T53,'(RECOMMENDED)')
0240      2      T20,'2 = NEW SSP AREA',T53,'(RECOMMENDED)')
0241      3      T20,'3 = USE HISTORICAL SSP',T53,'(RECOMMENDED)')
0242      4      T20,'4 = USE HISTORICAL DATA, '
0243      5      T20,'      USING THE LAYER (IF ANY)')
0244      6      T20,'      ASSOCIATED WITH YOUR BT '
0245      321      FORMAT(T20,'5 = USE XBT EXACTLY AS IS',
0246      1      T53,'(NOT RECOMMENDED BECAUSE '
0247      2/T53,'THE XBT HAS BEEN REJECTED '
0248      3/T53,'AND MAY PRODUCE UNRELIABLE '
0249      4/T53,'RESULTS'
0250      5///T20,'ENTER YOUR CHOICE ',T53,$)
0251      330      FORMAT(I3)
0252      340      FORMAT(' ENTER NEW SSP AREA',T50,$)
0253      END

```

## COMMAND QUALIFIERS

```
FORTRAN /CHECK=ALL/LIST/SHOW=(INCLUDE,NOMAP) [LAFLEUR]XBTErr.F77
```

```

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE_FORM)
/SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)
/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOB_LINES /NOCROSS_REFERENCE /N

```

## COMPILATION STATISTICS

```

Run Time:          3.25 seconds
Elapsed Time:      11.37 seconds
Page Faults:       412
Dynamic Memory:    158 pages

```

```

0001          SUBROUTINE XBTGRF(MONTH)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: XBTGRF
0005      ! AUTHOR: S. LAFLEUR & W. WACHTER, CODE 3333, NUSC/NLL
0006      ! DATE: 1982 & 12/83 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE XBTGRF ALLOWS THE OPERATOR TO PLOT
0008      !              ON A GRAPHICS TERMINAL.
0009      ! INPUTS: HARD COPY OPTION. PARAMETERS PASSED IN.
0010      !              VARIABLES IN COMMONS.
0011      ! OUTPUTS: CRT PROMPTING MESSAGES TO OPERATOR.
0012      ! MODULES CALLED: ICLR, INSERT, LAYER
0013      ! CALLED BY: XBT
0014      !
0015          INCLUDE 'GRF.INC'
0016      1 ! -----GRF-----
0017      1 ! VARBL   SIZE      PURPOSE                                TYPE      RANGE
0018      1 ! -----
0019      1 ! DBT      (25)     DEPTH OF DEPTH/VEL PAIR                REAL*4
0020      1 ! IANS                                           PREDICTION TYPE          INTEGER*2   -2 TO +2
0021      1 ! ILYR                                           INDEX FOR LAYER DEPTH    INTEGER*2
0022      1 ! INBT                                           OPERATOR ENTERED # OF BT POINTS  INTEGER*2
0023      1 ! ISVP                                           LATEST OR HISTORICAL BT FLAG    INTEGER*2   1 OR 2
0024      1 ! I2000                                          SVP INDEX FOR 2000 FT DEPTH      INTEGER*2
0025      1 ! VBT      (25)     VELOCITY FOR DEPTH PAIR REAL*4        REAL*4
0026      1
0027      1          REAL*4    DBT,VBT
0028      1          INTEGER*2 IANS, ILYR, INBT, ISVP, I2000
0029      1
0030      1          COMMON /GRF/ IANS, ISVP, ILYR, I2000, INBT, DBT(25), VBT(25)
0031      1
0032      1 ! -----END GRF-----
0033          INCLUDE 'OCEANS.INC'
0034      1
0035      1 ! -----OCEANS-----
0036      1 ! VARBL   SIZE      PURPOSE                                TYPE
0037      1 ! -----
0038      1 ! IOCEAN (50)     ARRAY OF NAMES OF OCEANS                DATA
0039      1
0040      1          INTEGER*2 IOCEAN
0041      1          DIMENSION IOCEAN(50)
0042      1
0043      1          DATA IOCEAN/'NO','RT','H ','PA','CI','FI','C ','OC','EA','N ',
0044      1          1 'NO','RT','H ','AT','LA','NT','IC','O','CE','AN',
0045      1          2 'ME','DI','TE','RR','AN','EA','N ','SE','A ',
0046      1          3 'IN','DI','AN','O','CE','AN',' ',
0047      1          4 'NO','RW','EG','IA','N ','SE','A ',
0048      1
0049      1          COMMON /OCEANS/ IOCEAN
0050      1
0051      1 ! -----END OCEANS-----
0052          INCLUDE 'SVP.INC'
0053      1 ! -----SVP-----
0054      1 ! VARBL   SIZE      PURPOSE                                TYPE      RANGE
0055      1 ! -----
0056      1 ! BDF                                           BOTTOM DEPTH IN FATHOMS      REAL*4
0057      1 ! BIOP                                           BIOLOGICAL BACK SCATTERING COEF  REAL*4
0058      1 ! BTDATE (9)     DATE OF LAST BT INPUT                BYTE
0059      1 ! BTTIME (8)     TIME OF LAST BT INPUT                BYTE

```

```

0060 1 ! C      (50)  VELOCITY (PAIRED WITH Z FOR SVP) REAL*4
0061 1 ! CC     (50)  VELOCITY (PAIRED WITH ZZ FOR SVP) REAL*4
0062 1 ! CS      SOUND VELOCITY AT SURFACE REAL*4
0063 1 ! DEG     TEMPERATURE (DEG) REAL*4 57.2957795
0064 1 ! EL      LAYER DEPTH DATA
0065 1 ! F       FREQUENCY REAL*4
0066 1 ! GRDS    GRIDS REAL*4 0.0164
0067 1 ! ITO     MINIMAL 2-WAY TRAVEL TIME INTEGER*2
0068 1 ! MGSOP   MGS PROVINCE NUMBER INTEGER*2
0069 1 ! N       # OF DEPTH/VELOCITY PAIRS INTEGER*2
0070 1 ! NN      # OF DEPTH/VELOCITY PAIRS INTEGER*2
0071 1 ! PI      MATHEMATICAL CONSTANT PI REAL*4 3.1415927
0072 1 ! SNDATE (9) DATE SYS PARMS LAST UPDATED BYTE
0073 1 ! SNTIME (8) TIME SYS PARMS LAST UPDTAED BYTE
0074 1 ! SYDATE (9) CURRENT DATE READ FROM SYSTEM BYTE
0075 1 ! SYTIME (8) CURRENT TIME READ FROM SYSTEM BYTE
0076 1 ! TMP      TEMPERATURE REAL*4
0077 1 ! UMKZ     BOTTOM BACK SCATTERING COEF. REAL*4 -28.0
0078 1 ! WS       WIND SPEED REAL*4
0079 1 ! Z        (50) DEPTH OF POINT OF SOUND SPEED REAL*4
0080 1 ! ZZ       (50) DEPTH OF POINT OF SOUND SPEED REAL*4
0081 1
0082 1          INTEGER*2 ITO,MGSOP,N,NN
0083 1          REAL*4    BDF,BIOP,C(50),CC(50),CS,DEG,EL,F,GRDS
0084 1          REAL*4    PI,TMP,UMKZ,WS,Z(50),ZZ(50)
0085 1          BYTE      SYDATE(9),SYTIME(8),BTDATE(9),BTTIME(8)
0086 1          BYTE      SNDATE(9),SNTIME(8)
0087 1          DATA     PI,DEG,GRDS/3.1415927,57.2957795,0.0164/
0088 1          DATA     UMKZ/-28./
0089 1
0090 1          COMMON /SVP/ F,N,Z,C,EL,MGSOP,BDF,WS,CS,TMP,BIOP,
0091 1          1          UMKZ,PI,DEG,GRDS,ITO,ZZ,CC,NN,
0092 1          2          SYDATE,SYTIME,BTDATE,BTTIME,SNDATE,SNTIME
0093 1 ! -----SVP-END-----
0094 1          INCLUDE 'SVPl.INC'
0095 1 ! -----SVPl-----
0096 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0097 1 ! -----
0098 1 ! BUFFER (224) HISTORICAL DATA FILE BUFFER REAL*4
0099 1 ! DS      (30)  HISTORICAL DEPTH REAL*4
0100 1 ! J20      # OF DEEP OCEAN DEPTH/VEL PAIRS INTEGER*2
0101 1 ! NS      TOTAL # OF PAIRS IN HISTORICAL INTEGER*2
0102 1 ! NSN     MONTH NUMBER (1=JAN.,ETC) INTEGER*2 1 TO 12
0103 1 ! SLNTY   SALINITY REAL*4
0104 1 ! VS      (30)  HISTORICAL VELOCITY REAL*4
0105 1
0106 1          REAL*4    BUFFER,DS,SLNTY,VS
0107 1          INTEGER*2 J20,NSN,NS
0108 1
0109 1          COMMON /SVPl/ J20,BUFFER(224),NSN,SLNTY,DS(30),VS(30),NS
0110 1 ! -----END SVPl-----
0111 1          INCLUDE 'LOC.INC'
0112 1 ! -----LOC-----
0113 1 ! VARBL  SIZE  PURPOSE                                TYPE  RANGE
0114 1 ! -----
0115 1 ! INDX      SSP INDEX INTEGER*2
0116 1 ! LAT      (4)  LATITUDE INTEGER*2
0117 1 ! LONG     (4)  LONGITUDE INTEGER*2
0118 1 ! NMAREA  (20)  AREA OCEAN NAME BYTE

```

```

0119 1 ! NOC                NUMBER OF OCEAN          INTEGER*2
0120 1 ! RCZ                RANGE TO CONVERG. ZONE REAL*4
0121 1
0122 1                REAL*4      RCZ
0123 1                INTEGER*2  INDX,LAT, LONG, NOC
0124 1                BYTE      NMAREA(20)
0125 1
0126 1                COMMON /LOC/ LAT(4),LONG(4),NOC,INDX,RCZ,NMAREA
0127 1
0128 1 ! -----END LOC-----
0129 1
0130 1 ! VARBL  SIZE      PURPOSE                                TYPE      RANGE
0131 1 ! -----
0132 1 ! DDLYR                LAYER DEPTH FOR SUBROUTINE CALL      REAL*4
0133 1 ! DTS      (50)        MONTHLY HISTORICAL DEPTH FROM BUFFER  REAL*4
0134 1 ! ENVEL                ENVELOPE FACTOR                      REAL*4
0135 1 ! I                    COUNTER                                INTEGER*2
0136 1 ! IC      (50)        VELOCITY IN RASTER UNITS              INTEGER*2
0137 1 ! ICOPY                NUMBER OF COPIES DESIRED BY OPERATOR  INTEGER*2
0138 1 ! IGTYP                DEEP PROFILE TYPE BASED ON DEPTH      INTEGER*2
0139 1 ! II                    COUNTER                                INTEGER*2
0140 1 ! IILYR                LAYER DEPTH FOR SUBROUTINE CALL      INTEGER*2
0141 1 ! IZ      (50)        DEPTH IN RASTER UNITS                  INTEGER*2
0142 1 ! J                    COUNTER                                INTEGER*2
0143 1 ! JJ                    FACTOR                                INTEGER*2
0144 1 ! JNBT                NUMBER OF BT                            INTEGER*2
0145 1 ! JVMIN                MINIMUM SOUND SPEED ON GRAPHS         INTEGER*2
0146 1 ! JVMAX                MAXIMUM SOUND SPEED ON GRAPHS        INTEGER*2
0147 1 ! K                    COUNTER                                INTEGER*2
0148 1 ! KNT                COUNTER FOR ENVELOPE                    INTEGER*2
0149 1 ! M                    POINTER FOR BUFFER ARRAY             INTEGER*2
0150 1 ! MONTH                MONTH SPECIFIED BY OPERATOR          INTEGER*2
0151 1 ! NSNS      (3)        STORE PREVIOUS,CURRENT, & NEXT MONTH  INTEGER*2
0152 1 ! NSN1                SEASON OF YEAR                        INTEGER*2
0153 1 ! N1                    NUMBER OF DEPTH/VEL PAIRS          INTEGER*2
0154 1 ! RMONTH (12)        NAMES OF MONTHS ARRAY                REAL*4
0155 1 ! SCHNLD                SOUND CHANNEL LAYER DEPTH           REAL*4
0156 1 ! THEBD                DEEP PROFILE GRAPH TYPE              REAL*4
0157 1 ! VMIN                MINIMUM SOUND SPEED                  REAL*4
0158 1 ! VTS      (50)        MONTHLY HISTORICAL VELOCITY FROM BUFFER REAL*4
0159 1 ! XMAX                GRAPHIC BOUNDARIES COORDS X AXIS MAX   REAL*4
0160 1 ! XMIN                GRAPHIC BOUNDARIES COORDS X AXIS MIN   REAL*4
0161 1 ! XX      (50)        VELOCITY IN RASTER UNITS              REAL*4
0162 1 ! YMAX                GRAPHIC BOUNDARIES COORDS Y AXIS MAX   REAL*4
0163 1 ! YMIN                GRAPHIC BOUNDARIES COORDS Y AXIS MIN   REAL*4
0164 1 ! YY      (50)        DEPTH IN RASTER UNITS                  REAL*4
0165 1
0166 1 ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0167
0168                INTEGER*2 I,IC,ICOPY,IGTYPE,II,IILYR,IZ,J
0169                INTEGER*2 JJ,JNBT,JVMIN,JVMAX,K,KNT,M,MONTH,NSNS,NSN1,N1
0170                REAL*4    DDLYR,DTS,ENVEL,RMONTH,SCHNLD,THEBD,VMIN
0171                REAL*4    VTS,XMAX,XMIN,XX,YMAX,YMIN,YY
0172
0173                DIMENSION IC(50),IZ(50),XX(50),YY(50)
0174                DIMENSION DTS(50),VTS(50)
0175                DIMENSION RMONTH(12),NSNS(3)
0176                DATA RMONTH /'JAN ','FEB ','MAR ','APR ','
0177                'MAY ','JUN ','JUL ','AUG ',

```

```

0178      2      'SEP ', 'OCT ', 'NOV ', 'DEC ' /
0179
0180 !-----PRELIMINARIES-----
0181      CALL ICLR      ! CLEAR SCREEN
0182      NSNS(1)=NSN-1  ! PREVIOUS MONTH
0183      IF(NSNS(1).EQ.0) NSNS(1)=12 ! START OF YEAR FIX
0184      NSNS(2)=NSN    ! CURRENT MONTH
0185      NSNS(3)=NSN+1  ! NEXT MONTH
0186      IF(NSNS(3).EQ.13) NSNS(3)=1 ! END OF YEAR FIX
0187
0188 !-----INITIALIZE SCREEN-----
0189      CALL INITT(3)   ! INITIALIZE TEKTRONIX 4025
0190      CALL SCREEN(100,600,100,400) ! DEFINE BOUNDS IN RASTERS
0191      XMIN=0.         ! GRAPHIC BOUNDARIES COORDS
0192      XMAX=300.       ! GRAPHIC BOUNDARIES COORDS
0193      YMIN=25000.     ! GRAPHIC BOUNDARIES COORDS
0194      YMAX=0.         ! GRAPHIC BOUNDARIES COORDS
0195      CALL UWINDO(XMIN,XMAX,YMIN,YMAX) ! DEFINE BOUNDS IN USER UNITS
0196
0197 !-----DRAW BOXES-----
0198      CALL MOVEU(XMIN,YMIN) ! MOVE BEAM TO THESE COORDS
0199      CALL DRAWU(XMAX,YMIN) ! DRAW VECTOR TO THESE COORDS
0200      CALL DRAWU(XMAX,(YMAX+1600.)) ! DRAW VECTOR TO THESE COORDS
0201      CALL DRAWU(XMAX-XMAX/3.,(YMAX+1600.)) ! DRAW VECTOR
0202      CALL DRAWU(XMAX-XMAX/3.,YMIN) ! DRAW VECTOR TO THESE COORDS
0203      CALL MOVEU(XMAX-XMAX/3.,(YMAX+1600.)) ! MOVE BEAM TO COORDS
0204      CALL DRAWU(XMAX-XMAX/3.*2.,(YMAX+1600.)) ! DRAW VECTOR
0205      CALL DRAWU(XMAX-XMAX/3.*2.,YMIN) ! DRAW VECTOR TO THESE COORDS
0206      CALL MOVEU(XMAX-XMAX/3.*2.,(YMAX+1600.)) ! MOVE BEAM
0207      CALL DRAWU(XMIN,(YMAX+1600.)) ! DRAW VECTOR TO THESE COORDS
0208      CALL DRAWU(XMIN,YMIN) ! DRAW VECTOR TO THESE COORDS
0209
0210 !-----LABEL BOXES-----
0211      CALL SYMBOL(0,300,1,14,'D E P T H F T') ! PLOT WORDS
0212      CALL MOVEU(XMIN+50.,YMAX-1400.) ! MOVE BEAM TO THESE COORDS
0213      CALL TEXT(4,'LAST') ! PLOT STRING TEXT
0214      CALL MOVEU(XMIN+130.,YMAX-1400.) ! MOVE BEAM TO THESE COORDS
0215      CALL TEXT(7,'CURRENT') ! PLOT STRING TEXT
0216      CALL MOVEU(XMIN+240.,YMAX-1400.) ! MOVE BEAM TO THESE COORDS
0217      CALL TEXT(4,'NEXT') ! PLOT STRING TEXT
0218      CALL MOVEU(XMIN+30.,YMAX-400.) ! MOVE BEAM TO THESE COORDS
0219      CALL TEXT(7,'MONTH [') ! PLOT STRING TEXT
0220      CALL MOVEU(XMIN+60.,YMAX-400.) ! MOVE BEAM TO THESE COORDS
0221      CALL INUMBR(NSNS(1),2) ! CONVERT INTEGER*2 TO ASCII
0222      CALL MOVEU(XMIN+70.,YMAX-400.) ! MOVE BEAM TO THESE COORDS
0223      CALL TEXT(1,']') ! PLOT STRING TEXT
0224      CALL MOVEU(XMIN+130.,YMAX-400.) ! MOVE BEAM TO THESE COORDS
0225      CALL TEXT(7,'MONTH [') ! PLOT STRING TEXT
0226      CALL MOVEU(XMIN+163.,YMAX-400.) ! MOVE BEAM TO THESE COORDS
0227      CALL INUMBR(NSNS(2),2) ! CONVERT INTEGER*2 TO ASCII
0228      CALL MOVEU(XMIN+171.,YMAX-400.) ! MOVE BEAM TO THESE COORDS
0229      CALL TEXT(1,']') ! PLOT STRING TEXT
0230      CALL MOVEU(XMIN+230.,YMAX-400.) ! MOVE BEAM TO THESE COORDS
0231      CALL TEXT(7,'MONTH [') ! PLOT STRING TEXT
0232      CALL MOVEU(XMIN+263.,YMAX-400.) ! MOVE BEAM TO THESE COORDS
0233      CALL INUMBR(NSNS(3),2) ! CONVERT INTEGER*2 TO ASCII
0234      CALL MOVEU(XMIN+271.,YMAX-400.) ! MOVE BEAM TO THESE COORDS
0235      CALL TEXT(1,']') ! PLOT STRING TEXT
0236      CALL MOVEU(XMIN-30.,YMAX+1550.) ! MOVE BEAM TO THESE COORDS

```

```

0237      CALL TEXT(1,'0')          ! PLOT STRING TEXT
0238      CALL MOVEU(XMIN-30.,13000.) ! MOVE BEAM TO THESE COORDS
0239      CALL TEXT(4,'1000')        ! PLOT STRING TEXT
0240      CALL MOVEU(XMIN-30.,YMIN)   ! MOVE BEAM TO THESE COORDS
0241      CALL TEXT(4,'2000')        ! PLOT STRING TEXT
0242      CALL MOVEU(XMIN+50.,YMIN+1000.) ! MOVE BEAM TO THESE COORDS
0243      CALL TEXT(3,RMONTH(NSNS(1))) ! PLOT STRING TEXT
0244      CALL MOVEU(XMIN+150.,YMIN+1000.) ! MOVE BEAM TO THESE COORDS
0245      CALL TEXT(3,RMONTH(NSNS(2))) ! PLOT STRING TEXT
0246      CALL MOVEU(XMIN+250.,YMIN+1000.) ! MOVE BEAM TO THESE COORDS
0247      CALL TEXT(3,RMONTH(NSNS(3))) ! PLOT STRING TEXT
0248      CALL MOVEU(XMIN+50.,YMIN+3000.) ! MOVE BEAM TO THESE COORDS
0249      IF(NOC.EQ.1) CALL TEXT(20,IOCEAN) ! DISPLAY NPAC OCEAN
0250      IF(NOC.EQ.2) CALL TEXT(20,IOCEAN(11)) ! DISPLAY NLANT OCEAN
0251      IF(NOC.EQ.3) CALL TEXT(20,IOCEAN(21)) ! DISPLAY MED SEA
0252      IF(NOC.EQ.4) CALL TEXT(20,IOCEAN(31)) ! DISPLAY INDIAN OCEAN
0253      IF(NOC.EQ.5) CALL TEXT(20,IOCEAN(41)) ! DISPLAY NORWEGIAN SEA
0254      CALL MOVEU(XMIN+226.,YMIN+3000.) ! MOVE BEAM TO THESE COORDS
0255      CALL TEXT(9,'SSP AREA:')      ! PLOT STRING TEXT
0256      CALL MOVEU(XMIN+298.,YMIN+3000.) ! MOVE BEAM TO THESE COORDS
0257      CALL INUMBR(INDX,2)          ! DISPLAY SSP INDEX #
0258      CALL MOVEU(XMIN+50.,YMIN+4000.) ! MOVE BEAM TO THESE COORDS
0259      CALL TEXT(38,'*** RAW BT DATA IS SHOWN WITH X'S ***') ! PLOT
0260      CALL MOVEU(XMIN+25.,YMIN+5000.) ! MOVE BEAM TO THESE COORDS
0261      CALL TEXT                    ! PLOT STRING TEXT
0262      1      (53,'*** HISTORICAL SSP DATA IS SHOWN WITH SOLID LINES ***')
0263      II=1                          ! INITIALIZE COUNTER
0264
0265      !-----DRAW WITHIN BOUNDARIES-----
0266      DO 777 MONTH=100,434,167      ! DO FOR GRAPHIC AREA
0267      CALL UNCLIP                    ! DISABLE CLIPPING BY DRAW
0268      IF(MONTH.EQ.100)CALL CLIP(100,266,100,400) ! DEFINE CLIP -
0269      IF(MONTH.EQ.267)CALL CLIP(266,432,100,400) ! DRAW NO LINES
0270      IF(MONTH.EQ.434)CALL CLIP(432,600,100,400) ! OUTSIDE CLIP
0271      NSN1=NSNS(II)                 ! SEASON NUMBER
0272      II=II+1                       ! INCREMENT COUNTER
0273
0274      !-----EXTRACT THE MONTHLY HISTORICAL D
0275      M=(NSN1-1)*14+1               ! POINTER FOR BUFFER ARRAY
0276      DO 110 J=1,7                  ! DO FOR 1 TO 7
0277      DTS(J)=BUFFER(M)              ! DEPTH FROM BUFFER
0278      VTS(J)=BUFFER(M+1)            ! VELOCITY FROM BUFFER
0279      IF(J.GT.1.AND.DTS(J).LE.1.) GO TO 120 ! NEGATIVE DEPTH
0280      M=M+2                          ! INCREASE POINTER VALUE
0281      110      CONTINUE              ! END DO LOOP
0282      J=8                            ! SET J FOR NEXT DO LOOP
0283      120      N1=J20+J-1            ! NUMBER OF DEPTH/VEL PAIRS
0284      M=169                          ! POINTER FOR BUFFER ARRAY
0285      DO 130 K=J,N1                 ! DO FROM 8 TO # OF PAIRS
0286      DTS(K)=BUFFER(M)              ! DEPTH FROM BUFFER
0287      VTS(K)=BUFFER(M+1)            ! VELOCITY FROM BUFFER
0288      M=M+2                          ! INCREASE POINTER VALUE
0289      130      CONTINUE              ! END DO LOOP
0290
0291      !-----DETERMINE DEEP PROFILE GRAPH TYPE BY TRUE BOTTOM
0292      THEBD=BDF*6.0                 ! DEEP PROFILE GRAPH TYPE
0293      IF(THEBD.GE.0..AND.THEBD.LE.12000.) IGTYP=1 ! TYPE ONE
0294      IF(THEBD.GT.12000..AND.THEBD.LE.16000.) IGTYP=2 ! TYPE TWO
0295      IF(THEBD.GT.16000.) IGTYP=3 ! TYPE THREE

```

```

0296
0297 !-----DETERMINE THE MINIMUM SOUND SPEED, SOUND CHANNEL DEPTH
0298 SCHNLD=10000. ! SOUND CHANNEL LAYER DEPTH
0299 VMIN=VTS(1) ! MINIMUM SOUND SPEED
0300 CALL LAYER(N1,DTS,VTS,DDLyr) ! DEPTH OF SURFACE DUCT LAYER
0301 CALL INSERT(N1,DTS,VTS,DDLyr,IILyr) ! INSERT POINT INTO SVP
0302
0303 DO 53 I=IILyr+1,N1 ! DO FROM LAYER TO # OF PAIRS
0304 IF(VTS(I).LE.VTS(I+1).AND.SCHNLD.EQ.10000.)
0305     1 SCHNLD=DTS(I) ! SOUND CHANNEL LAYER DEPTH
0306 IF(VTS(I).LT.VMIN)VMIN=VTS(I) ! SET MIN VELOCITY
0307 53 CONTINUE ! END DO LOOP
0308 JJ=(VMIN-10.)/50. ! FACTOR FOR JVMIN
0309 JVMIN=50*JJ ! MIN VELOCITY
0310 JVMAX=JVMIN+300 ! MAX VELOCITY
0311 KNT=1 ! SET ENVELOPE COUNTER
0312
0313 !-----CONVERT Z'S AND C'S TO RASTER UNITS AND DRAW DE
0314 200 DO 230 K=1,N1 ! DO FOR NUMBER OF PAIRS
0315 XX(K)=.64*(VTS(K)-JVMIN)+MONTH ! .64=RASTERS/(FT/SEC)
0316 YY(K)=343-(.14*DTS(K))+37. ! 343=RASTER HT OF 0*DEPTH
0317 IC(K)=XX(K) ! VELOCITY IN RASTERS
0318 IZ(K)=YY(K) ! DEPTH IN RASTERS
0319 230 CONTINUE ! END DO LOOP
0320 CALL CONECT(IC(1),IZ(1),IC(2),IZ(2)) ! DRAW LINE
0321 DO 240 M=3,N1 ! # OF DEPTH/VEL PAIRS
0322 CALL DRAW(IC(M),IZ(M)) ! DRAW BEAM WITHIN CLIP
0323 240 CONTINUE ! END DO LOOP
0324 DO 250 K=1,N1 ! SET UP TO DRAW ENVELOPE
0325 ENVEL=15.-.006*DTS(K) ! ENVELOPE
0326 IF(KNT.LE.1) THEN ! FIRST TIME THROUGH
0327 VTS(K) = VTS(K) - ENVEL ! VELOCITY ENVELOPE
0328 ELSE ! SECOND, THIRD TIME THROUGH
0329 VTS(K) = VTS(K) + 2*ENVEL ! VELOCITY ENVELOPE
0330 END IF ! END IF BLOCK
0331 250 CONTINUE ! END DO LOOP
0332 KNT = KNT + 1 ! INCREASE COUNT
0333 IF(KNT.LE.3) GO TO 200 ! GO BACK FOR 2ND AND 3RD TIMES
0334
0335 !-----PLOT THE RAW BT DATA-----
0336 IF(INBT.GT.0) THEN ! IF = 0 GO TO HARDCOPY OPTION
0337 DO 700 M=1,INBT ! DO FOR # OF BTS
0338 IF(DBT(M).GT.2000.0) GO TO 193 ! BOTTOM DEPTH > 2000'
0339 XX(M)=.64*(VBT(M)-JVMIN)+MONTH ! .64=RASTERS/(FT/SEC)
0340 IC(M)=XX(M) ! VELOCITY
0341 YY(M)=343-(.14*DBT(M))+37. ! 343=RASTER HT OF 0*DEPTH
0342 IZ(M)=YY(M) ! DEPTH
0343 700 CONTINUE ! END DO LOOP
0344 193 JNBT=M-1 ! NUMBER OF BT
0345 DO 194 K=1,2 ! DO TWICE
0346 CALL SYMBOL((IC(K)-2),(IZ(K)-2),0,1,'X') ! PLOT WORDS
0347 IF(INBT.EQ.1) GO TO 777 ! FIRST BT
0348 194 CONTINUE ! END DO LOOP
0349 IF(JNBT.GE.3) THEN ! IF MORE THAN OR = THREE
0350 DO 195 M=3,JNBT ! DO FOR # OF BT
0351 CALL SYMBOL((IC(M)-2),(IZ(M)-2),0,1,'X') ! PLOT WORDS
0352 195 CONTINUE ! END DO LOOP
0353 END IF ! END IF
0354 CALL LNTYPE(2) ! LINE TYPE .....

```



```

0355          CALL CONECT(IC(1),IZ(1),IC(2),IZ(2)) ! DRAW LINE
0356          DO 198 M=3,JNBT                      ! DO FOR NUMBER OF BTS
0357          CALL DRAW(IC(M),IZ(M))                ! DRAW BEAM WITHIN CLIP
0358 198      CONTINUE                              ! END DO LOOP
0359          CALL LNTYPE(1)                        ! LINE TYPE
0360          END IF                                ! END IF BLOCK
0361 777      CONTINUE                              ! END DO LOOP
0362          CALL MOVE(0,0)                        ! DUMMY CALL TO DELAY EXIT
0363
0364  !-----HARDCOPY OPTION-----
0365 778      WRITE(5,780)                          ! RING BELL FOR ATTENTION
0366          WRITE(5,781)                          ! ENTER MONTH PROMPT
0367          READ(5,787) MONTH                      ! OPERATOR'S RESPONSE
0368          IF(MONTH.NE.NSNS(1).AND.MONTH.NE.NSNS(2).AND.MONTH.NE.NSNS(3))GO
0369          WRITE(5,800)                          ! NUMBER OF COPIES QUERY
0370          READ(5,787) ICOPY                      ! # OF COPIES NEEDED
0371          IF(ICOPY.NE.0) THEN                    ! COPIES WANTED
0372              DO 900 I=1,ICOPY                  ! DO FOR # OF COPIES
0373              WRITE(5,1000)                     ! PRINT SCREEN IMAGE
0374 900      CONTINUE                              ! END DO LOOP
0375          END IF                                ! END IF BLOCK
0376          CALL UNCLIP                            ! NO CLIPPING
0377          CALL ICLR                             ! CLEAR SCREEN
0378
0379  !-----FORMAT STATEMENTS-----
0380 780      FORMAT(100('/' !BEL'))
0381 781      FORMAT(1H$, ' ENTER MONTH# YOU THINK MATCHES THE XBT BEST',
0382 1          T60,' ')
0383 787      FORMAT(I2)
0384 800      FORMAT(1H$, ' HOW MANY HARD COPIES WOULD YOU LIKE? [0,1,2,ETC.]',
0385 1          T60,' ')
0386 1000     FORMAT(' !HCO S')
0387
0388          RETURN
0389          END

```

```

0001          SUBROUTINE XBTMOD(NBT,NI,N2,NDLYR)
0002
0003      ! PROLOGUE:
0004      ! MODULE NAME: LYRMOD
0005      ! AUTHOR: S. LAFLEUR, W. WACHTER (FORTRAN 77)
0006      ! DATE: 7/84 & 7/84 (FORTRAN 77)
0007      ! FUNCTION: SUBROUTINE XBTMOD LIMITS THE XBT SOUND SPEEDS
0008      !              TO THE TOLERANCE ENVELOPE ABOUT THE HISTORICAL
0009      !              DATA. IF THE LAST PART OF THE XBT IS OUT OF
0010      !              TOLERANCE, THESE 'NOUT' POINTS WILL BE DELETED.
0011      !              IF THIS CAUSES THE LAYER DEPTH POINT TO BE DELETED,
0012      !              THE LAST GOOD POINT BECOMES THE NEW LAYER DEPTH.
0013      !              DURING THE SECOND EXECUTION OF THIS SUBROUTINE,
0014      !              THE XBT DATA IS EXTENDED TO 2500 FEET BY CALLING 'ASIS'.
0015      ! INPUTS: PARAMETERS PASSED IN & VARIABLES IN COMMONS.
0016      ! OUTPUTS: MODIFY SS TO STAY WITHIN TOLERANCE ENVELOPE
0017      ! MODULES CALLED: ASIS
0018      ! CALLED BY: XBT
0019      !
0020          INCLUDE 'DTV.INC'
0021      1 ! -----DTV-----
0022      1 !  VARBL  SIZE  PURPOSE                                TYPE  RANGE
0023      1 !  -----  ----  -----                                ----  -----
0024      1 !  D      (25)  DEPTH                                    REAL*4
0025      1 !  DD     (25)  DEPTH                                    REAL*4
0026      1 !  NNBT                                NUMBER OF BATHETHERMAL  INTEGER*2
0027      1 !  T      (25)  TEMPERATURE                            REAL*4
0028      1 !  TT     (25)  TEMPERATURE                            REAL*4
0029      1 !  VEL    (25)  VELOCITY                                REAL*4
0030      1 !
0031      1 !              INTEGER*2 NNBT
0032      1 !              REAL*4 D,DD,T,TT,VEL
0033      1 !
0034      1 !              COMMON /DTV/  D(25),T(25),VEL(25),DD(25),TT(25),NNBT
0035      1 ! -----END DTV-----
0036          INCLUDE 'SVPl.INC'
0037      1 ! -----SVPl-----
0038      1 !  VARBL  SIZE  PURPOSE                                TYPE  RANGE
0039      1 !  -----  ----  -----                                ----  -----
0040      1 !  BUFFER (224)  HISTORICAL DATA FILE BUFFER          REAL*4
0041      1 !  DS      (30)  HISTORICAL DEPTH                      REAL*4
0042      1 !  J20                                # OF DEEP OCEAN DEPTH/VEL PAIRS  INTEGER*2
0043      1 !  NS                                TOTAL # OF PAIRS IN HISTORICAL  INTEGER*2
0044      1 !  NSN                                MONTH NUMBER (1=JAN.,ETC)        INTEGER*2  1 TO 12
0045      1 !  SLNTY                                SALINITY                            REAL*4
0046      1 !  VS      (30)  HISTORICAL VELOCITY                    REAL*4
0047      1 !
0048      1 !              REAL*4  BUFFER,DS,SLNTY,VS
0049      1 !              INTEGER*2 J20,NSN,NS
0050      1 !
0051      1 !              COMMON /SVPl/ J20,BUFFER(224),NSN,SLNTY,DS(30),VS(30),NS
0052      1 ! -----END SVPl-----
0053      !
0054      !  VARBL  SIZE  PURPOSE                                TYPE  RANGE
0055      !  -----  ----  -----                                ----  -----
0056      !  DE                                DEPTH                                    REAL*4
0057      !  I                                LOOP COUNTER                            INTEGER*2
0058      !  K                                LOOP COUNTER                            INTEGER*2
0059      !  NBT                                NUMBER OF BT POINTS                    INTEGER*2

```

```

0060      ! NDLYR          BT LAYER'S POSITION IN ARRAY      INTEGER*2
0061      ! NI            # OF HISTORICAL POINTS           INTEGER*2
0062      ! NOUT          # OF POINTS OUTSIDE ENVELOPE      INTEGER*2
0063      ! N2            FLAG FOR SECOND TIME THRU        INTEGER*2
0064      ! XX            ENVELOPE TOLERANCE WIDTH         REAL*4
0065      !
0066      ! *** VARIABLES NOT LISTED HERE SHOULD APPEAR IN COMMONS ***
0067
0068      INTEGER*2 I,K,NBT,NDLYR,NI,NOUT,N2
0069      REAL*4      DE,XX
0070
0071      !-----PRELIMINARIES-----
0072      NOUT=0          ! INIT # POINTS OUTSIDE ENVELOPE
0073      DO 70 I=1,NBT  ! DO FOR # OF BT POINTS
0074      IF(D(I).GE.2500.) GO TO 75 ! DEPTH > 2500 FEET
0075      DO 20 K=1,NI    ! FOR HISTORICAL POINTS
0076      IF(DS(K).LE.D(I).AND.DS(K+1).GT.D(I)) GO TO 30 ! DEPTH
0077      20 CONTINUE    ! END DO LOOP
0078      K=NI            ! DEFINE K
0079      30 DE=VS(K)+((VS(K+1)-VS(K))/(DS(K+1)-DS(K)))*(D(I)-DS(K)) ! DEP
0080
0081      !-----LIMIT XBT VELOCITIES TO ENVELOPE-----
0082      XX=15.-.006*D(I) ! ENVELOPE WIDTH
0083      IF(DE+XX.LT.VEL(I)) THEN ! OUTSIDE ENVELOPE
0084      VEL(I)=DE+XX      ! RESET VELOCITY
0085      NOUT=NOUT+1      ! INCREMENT # OUTSIDE ENVELOPE
0086      ELSE              ! INSIDE ENVELOPE
0087      IF(DE-XX.GT.VEL(I)) THEN ! OUTSIDE ENVELOPE
0088      VEL(I)=DE-XX      ! RESET VELOCITY
0089      NOUT=NOUT+1      ! INCREMENT # OUTSIDE ENVELOPE
0090      ELSE              ! INSIDE ENVELOPE
0091      NOUT=0            ! WITHIN ENVELOPE
0092      ENDIF            ! END IF BLOCK
0093      ENDIF            ! END IF BLOCK
0094      70 CONTINUE      ! END DO LOOP
0095      GO TO 80          ! DO NOT RESET # OF BT
0096      75 NBT = I-1      ! RESET NUMBER OF BT
0097      80 IF(NOUT.GT.0.AND.NDLYR.GT.NBT-NOUT) NDLYR=NBT-NOUT ! RESET NDLYR
0098      NBT=NBT-NOUT      ! RESET NUMBER OF BT POINTS
0099      IF(N2.EQ.0) CALL ASIS (NBT,D,VEL) ! IF XBT DATA EXTEND TO 2500'
0100      RETURN          ! RETURN TO CALLING ROUTINE
0101      END              ! END SUBROUTINE

```